Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Blaž Divjak

Enovita infrastruktura za nastavljanje in upravljanje naprav in storitev v omrežju

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: dr. Andrej Brodnik

Ljubljana, 2016

Zahvala

Zahvaljujem se mentorju dr. Andreju Brodniku za strokovno vodenje in usmerjanje ter spodbudo pri izdelavi magistrskega dela. Zahvale gredo tudi strokovnjakom z Arnesa, ki so mi omogočili izdelavo magistrskega dela.

Iskrena zahvala gre družini in punci Maši, ki so mi stali ob strani in me podpirali.

Blaž Divjak, 2016

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ACID	Atomicity, Consistency, Isolation, Durability	Atomarnost, Konsistentnost, Izolacija, Trajnost
ASCII	American Standard Code for Information Interchange	ameriški standardni nabor za izmenjavo informacij
ASN.1	Abstract syntax notation 1	Abstraktna sintaktična notacija 1
BER	Basic Encoding Rules	Osnovna pravila za kodiranje
CLI	Command line interface	Ukazna vrstica
CORBA	Common Object Request Broker Architecture	Arhitektura posrednikov zahtev skupnih objektov
CoAP	Constrained Application Protocol	Protokol za omejene aplikacije
CoMI	CoAP Management Interface	Vmesnik za upravljanje omejenih aplikacij
DMZ	Demilitarized zone	Demilitarizirana cona
GRNET	Greek Research and Technology Network	Grško raziskovalno in tehnološko omrežje
HTTP	HyperText Transfer Protocol	Protokol za prneos hiper teksta
HP	Hewlett Packard	Hewlett Packard
IAB	Internet Architecture Board	Odbor za internetno arhitekturo
IETF	Internet Engineering Task Force	Delovna skupina za medmrežni inženiring
IP	Internet protocol	medmrežni protokol
IPAM	IP address management	Upravljanje naslovnega prostora IP
ISO	Open Systems Interconnection	Odprt sistem za povezljivost

kratica	angleško	slovensko
JSON	JavaScript object notation	Notacija objektov JavaScript
MDB	Management database	Upravljalska baza
MIB	Management information base	Baza s podatki o objektih na napravi
NAPALM	Network Automation and Programmability Abstraction Layer with Multivendor support	Abstrakcijski nivo za avtomatizacijo in programabilnost omrežij s podporo za različne proizvajalce
NETCONF	Network configuration protocol	Protokol za nastavljanje omrežij
OID	Object ID	Enolični identifikator objekta
OVSDB	Open virtual switch database management protocol	Upravljavski protokol baze odprtega navideznega stikala
REST	Representational State Transfer	Predstavitev stanj prenosa spletnih zahtev
RFC	Request for comments	Zahtevek za komentarje
RPC	Remote procedure call	Klic oddaljene procedure
SDN	Software defined networking	Programsko definirana omrežja
SMI	Structure of Management Information	Struktura upravljalske informacije
SNMP	Simple network management protocol	Preprost protokol za upravljanje omrežij
SOAP	Simple Object Access Protocol	Preprost protoko za dostop do objektov
SSH	Secure Shell	Protokol za upravljanje računalnika na daljavo
TCP	Transmission Control Protocol	Protokol za nadzor prenosa
TLV	type-length-value	tip-dolžina-vrednost

kratica	angleško	slovensko
UDP	User Datagram Protocol)	Protokol za prenašanje datagramov
USM	User-based security model	Uporabniško naravnan varnostni model
URI	Uniform Resource Identifier	Uniformni identifikator virov
VACM	View-based access model	Model za omejevanje dostopa do informacij
WADL	Web Application Description Language	Jezik za opis spletnih storitev
XML	Extensible Markup Language	Razširljivi označevalni jezik
YANG	Yet Another Next Generation	Še ena naslednja generacija

Povzetek

Naslov: Enovita infrastruktura za nastavljanje in upravljanje naprav in storitev v omrežju

Računalništvo, ki je vse bolj samopostrežno naravnano, od storitev zahteva, da se prilagajajo uporabniku. S tem so se spremenile tudi zahteve za upravljanje in nastavljanje omrežne infrastrukture. Upravljalci so zaradi pogostih sprememb v omrežjih postavljeni pred izziv, kako čimbolj učinkovito preiti na programsko upravljanje računalniških omrežij in storitev.

V magistrskem delu smo preučili področje upravljanja in nastavljanja računalniških omrežij in storitev. Pri tem se osredotočili na pristope k upravljanju, protokole za upravljanje omrežnih elementov ter jezike za modeliranje podatkov o njihovih nastavitvah in operativnem stanju. Raziskali smo obstoječe rešitve za upravljanje in jih razdelili v štiri skupine.

Za vpeljavo programskega upravljanja ter nastavljanja omrežnih naprav in storitev smo predlagali arhitekturo enovite infrastrukture. Arhitekturo smo implementirali in preizkusili na laboratorijskem omrežju. Na dveh primerih, s katerima se srečujejo upravljalci omrežij, smo demonstrirali uporabnost predlagane infrastrukture za nastavljanje omrežnih naprav in storitev ter za avtomatsko odzivanje na težave v omrežju.

Ključne besede

netconf, yang, snmp, jinja2, rest, json, xml, avtomatizacija, upravljanje omrežij, storitve, ansible, openconfig, python, django, elastic stack

Abstract

Title: Uniform infrastructure for configuration and management of network devices and services

Today computer landscape is service orientated and services need to adapt to user's wishes. This has also changed the requirements for management and configuration of the network infrastructure. Due to frequent changes in the network operators are faced with the challenge of how to most effectively implement programmatic management of computer networks and services.

In this master thesis, we studied management and configuration of computer networks and services. We put emphasis on different approaches and protocols for managing network elements and languages used to model their configuration and operating state. We surveyed existing solutions for network management and placed them into four groups.

We proposed architecture for uniform network device and service management and configuration. We implemented and tested proposed architecture in our laboratory testbed network. We used two common use cases network managers face today to demonstrate its usefulness. In doing so we showed how network devices and services could be configured and how problems in the network could be remediated using the proposed uniform infrastructure.

Keywords

netconf, yang, snmp, jinja2, rest, json, xml, automatization, network management, services, ansible, openconfig, python, django, elastic stack

Poglavje 1

Uvod

Statično arhitekturo tradicionalnih omrežij je zamenjala dinamična, razširljiva in močno prilagodljiva. Spremembe narekuje hiter razvoj računalništva, ki je vse bolj samopostrežno naravnano in od storitev zahteva, da se prilagajajo uporabniku. Prilagodljivost storitev spreminja dinamiko omrežne infrastrukture in vpliva na to, kako pogosto se spreminjajo nastavitve na omrežnih napravah in storitvah. Zaradi vse večje potrebe po povezljivosti so računalniška omrežja postala večja in kompleksnejša. Sestavljena so iz velikega števila heterogenih omrežnih naprav, ki opravljajo različne naloge. Storitve omrežne infrastrukture pogosto tečejo tudi na strežnikih, kar zabriše meje med omrežno in strežniško infrastrukturo. Omrežne naprave se razlikujejo v funkcionalnosti, operacijskih sistemih in podpori protokolom. Največjo težavo pri nastavljanju in upravljanju omrežnih naprav predstavljajo različni upravljalski vmesniki in različna predstavitev podatkov o nastavitvah in podatkov o opративnem stanju. Na njihovo standardizacijo se zaradi različnih komercialnih vzgibov proizvajalci dolga leta niso ozirali.

Spremenjena dinamika prilagajanja infrastrukture je zahtevala nove pristope k upravljanju omrežij, saj tradicionalni pristop, pri katerem je upravljalec z uporabo enega upravljalskega protokola upravljal enega ali več omrežnih elementov, zaradi naštetih dejavnikov ni več primeren. Prvi pristop, ki se je pojavil, se imenuje programsko definirana omrežja *(SDN - Software defined*

networking). Arhitektura programsko definiranih omrežij razširja arhitekturo tradicionalnih in dodaja boljšo podporo za dinamično prilagajanje in razširjanje omrežne infrastrukture. Definira ločevanje krmilne in podatkovne ravnine omrežnega prometa. Njihovo upravljanje se izvaja s pomočjo krmilnikov SDN. Primer takega krmilnika je OpenDaylight [4]. Drugi pristop k upravljanju vključuje koncepte, prenešene s strežniške storitvene infrastrukture. Tam trende narekujejo orodja za avtomatizacijo upravljanja in nastavljanja. Upravljanje in nastavljanje potekata z uporabo programske opreme, ki centralizirano upravlja strežnike in storitve, ki tečejo na njih. Glavna predstavnika sta orodji Puppet [46] in Ansible [27].

Z namenom zmanjšati različnost upravljalskih vmesnikov in bolje podpreti nove pristope k upravljanju je bilo razvitih nekaj novih protokolov za upravljanje. Primer protokola, s katerim želi skupnost izboljšati postopek upravljanja omrežnih elementov, je NETCONF [18]. Za modeliranje enotne predstavitve podatkov o nastavitvah in operativnem stanju omrežnih elementov je skupnost razvila jezik YANG [16]. NETCONF in YANG izboljšujeta pomanjkljivosti dveh metod, ki se v tradicionalni arhitekturi računalniških omrežij pogosto uporabljata za upravljanje omrežne infrastrukture, to sta SNMP in ukazna vrstica (*ang. Command line interface*) [21]. Kljub izboljšavam, ki jih NETCONF in YANG prinašata, na mnogih omrežnih napravah še nista podprta.

Postavimo se v vlogo upravljalca večje omrežne infrastrukture, ki se razprostira čez širše geografsko območje. Njegova omrežna infrastruktura je sodobna, neprestano se širi in njene nastavitve se pogosto spreminjajo. Infrastrukturo sestavlja veliko število omrežnih naprav, nekatere storitve pa tečejo na strežnikih z operacijskim sistemom Linux. Svoj sistem za upravljanje in nastavljanje omrežne infrastrukture želi izboljšati. Rad bi, da je sistem pripravljen na dinamično omrežno infrastrukturo, s kakršno se srečujemo danes. Njegov glavni cilj je z novim sistemom poenostaviti upravljanje in nastavljanje omrežnih naprav in storitev, ki jih ponuja svojim uporabnikom. Njegova zahteva je tudi, da njegovi človeški operaterji in aplikacije ne bodo več

spreminjali nastavitev neposredno na napravah ali strežnikih, ampak bodo želeno stanje omrežne infrastrukture sporočili sistemu. Sistem bo nato sprožil ustrezne akcije, da se bodo spremembe uveljavile na omrežnih napravah in storitvah. V preteklosti je imel namreč več slabih izkušenj z izpadi storitev, ki so se pripetili zaradi napak pri neposrednem spreminjanju nastavitev s strani njegovih omrežnih tehnikov. Pri načrtovanju sistema se upravljalec sreča z nekaj temeljnimi vprašanji. Kakšen pristop za upravljanje izbrati? Ali obstaja čaroben protokol, s katerim bo lahko upravljal vse omrežne naprave in storitve? Kako doseči, da njegovim človeškim operaterjem ne bo več potrebno na pamet znati vseh ukazov za omrežna stikala?

V magistrskem delu bomo upravljalcu odgovorili na njegova vprašanja in mu pomagali razviti sistem za enovito upravljanje omrežnih naprav in storitev. Raziskali bomo področje upravljanja računalniških omrežij in storitev ter predlagali arhitekturo, ki bo nudila dobro izhodišče za vpeljavo dinamične, razširljive arhitekture računalniške infrastrukture. Za začetek v razdelku 1.1 identificiramo zahteve za sistem, ki bo upravljalcu pomagal pri doseganju njegovih ciljev. Naše strokovne prispevke predstavimo v razdelku 1.2, v razdelku 1.3 pa opišemo strukturo magistrskega dela.

1.1 Zahteve za rešitev

Po pregledu izbranega področja in določitvi raziskovalnega problema smo identificirali glavne cilje in zahteve za arhitekturo enovite infrastrukture za nastavljanje naprav in storitev v omrežju, ki bo upravljalcu pomagala doseči njegove cilje.

Logični model mora nuditi predstavitev omrežne infrastrukture in omogočati, da se nastavitve omrežnih naprav in storitev zapišejo v bazo. Model naj bo sestavljen iz dela, ki opisuje jedrne nastavitve, potrebne za delovanje omrežja, in iz dela, kjer bodo opisane nastavitve omrežnih storitev. Nabor podprtih storitev mora biti razširljiv, saj želi sčasoma upravljalec podpreti

vse storitve, katerih nastavitve se pogosto spreminjajo. Vse spremembe nastavitev omrežnih naprav in storitev, ki jih bodo zahtevali uporabniki bodisi preko grafičnih uporabniških vmesnikov bodisi skript, se morajo zapisati v bazo.

Programabilni vmesniki API so nujni za programabilni dostop do sistema. Vmesniki bodo omogočili spreminjanje nastavitev omrežne infrastrukture iz grafičnih vmesnikov in skript. To bo uporabnikom infrastrukture, omrežnim tehnikom in aplikacijam omogočilo, da vplivajo na želeno stanje omrežne infrastrukture, ki je zapisano v bazi, in posledično na njeno delovanje.

Abstrakcija mora omogočiti, da se podatki, shranjeni v podatkovni bazi, lahko pretvorijo v obliko, ki je podprta na širokem naboru omrežnih naprav. S tem bo zagotovljeno, da so podatki o nastavitvah in stanju omrežne naprave ali storitve neodvisni od proizvajalca (*ang. vendor agnostic*). Pri razvoju nivoja abstrakcije naj se, kjer je to mogoče, za predstavitev podatkov uporabi standardizirane podatkovne modele ali podatkovne modele, ki imajo široko podporo skupnosti.

Upravljanje omrežnih naprav in storitev mora vključevati podporo za upravljanje tako omrežnih naprav kot tudi strežnikov, na katerih tečejo storitve. V primeru, da nam med razvojem rešitve ne uspe odkriti čarobnega protokola, s katerim bi lahko upravljali vse omrežne naprave in strežnike, mora sistem vključevati možnost, da se za vsako napravo ali strežnik izbere najprimernejšega. To je tisti protokol, ki ga naprava najbolje podpira. Z uporabo upravljalskih protokolov mora sistem spremembe, zapisane v bazi, dostaviti na omrežne naprave in strežnike samodejno in zanesljivo.

Dolgoročna uporabnost rešitve naj bo dosežena z uporabo standardiziranih in že razvitih rešitev, kjer je to mogoče. To bo omogočilo črpanje novih

funkcionalnosti iz prispevkov skupnosti in industrije. Obenem mora biti zasnova dovolj modularna in razširljiva, da bo omogočala preprosto dodajanje podpore za nove omrežne storitve.

1.2 Strokovni prispevki

V raziskavi smo se osredotočili na spreminjanje nastavitev omrežnih naprav in storitev. Glavni prispevki dela so predstavitev področja upravljanja računalniških omrežij in storitev. Raziskava aktualnih del in odprtih vprašanj, ki jih trenutno rešuje skupnost. Raziskava, predstavitev in ovrednotenje protokolov za upravljanje, ki so že v uporabi ali se šele standardizirajo. Obstoječe rešitve za upravljanje smo klasificirali v štiri kategorije in podrobno predstavili njihove predstavnike, ki smo jih uporabili v našem arhitekturnem modelu. Na podlagi opravljene raziskave smo razvili in preizkusili arhitekturni model za upravljanje omrežnih naprav in storitev. Preizkusili smo ga na dveh primerih uporabe, in sicer na primeru vzpostavitve omrežja in na primeru odzivanja na probleme v omrežju.

1.3 Struktura magistrske naloge

V poglavju 2 je predstavljena raziskava področja upravljanja omrežnih naprav in storitev. V izhodišču je pojasnjena terminologija. Opisani so različni pristopi k upravljanju omrežne infrastrukture. Z namenom ugotoviti, kakšne so možnosti za predstavitev upravljalske informacije, so predstavljeni jeziki, s katerimi lahko opišemo nastavitvene podatke in podatke o operativnem stanju omrežnih naprav. Nato so opisani protokoli in metode, ki se uporabljajo za upravljanje omrežnih naprav in storitev. Protokoli in metode so ovrednoteni na podlagi kazalnikov, ki smo jih določili v fazi analize. Kot iztočnica so podani pregled sorodnih del in odprta vprašanja, ki jih skupnost še rešuje.

Poglavje 3 obravnava obstoječe rešitve za upravljanje omrežij. Rešitve so klasificirane v štiri kategorije. Iz nabora rešitev smo izbrali tiste, ki smo jih

uporabili za implementacijo enovite infrastrukture za upravljanje in nastavljanje omrežnih naprav in storitev, in jih podrobno predstavili.

Poglavje 4 opisuje razvoj arhitekture infrastrukture za enovito upravljanje omrežnih naprav in storitev, kar je cilj magistrskega dela. Podrobno so predstavljeni sestavni deli implementacije infrastrukture. Prikazan je priporočen pristop za umestitev infrastrukture v obstoječe omrežje in podan predlog postopka za nadaljni razvoj storitev.

V poglavju 5 smo ovrednotili predlagano infrastrukturo na podlagi zahtev, definiranih v razdelku 1.1. Predstavljeno je laboratorijsko omrežje, ki smo ga z namenom preizkušanja tehnologij in ovrednotenja infrastrukture vzpostavili na Akademski in raziskovalni mreži Slovenije. Prikazano je delovanje sistema na dveh primerih. Na primeru razširitve omrežne infrastrukture je prikazana nastavitev storitve in omrežnih elementov na eni izmed organizacij. Na drugem primeru je predstavljeno, kako se lahko enovita infrastruktura uporabi za odziv na težave v omrežju. Na podlagi izkušenj, pridobljenih med preizkušanjem, so podana izhodišča za nadaljnji razvoj.

V poglavju 6 so zapisane sklepne ugotovitve.

Poglavje 2

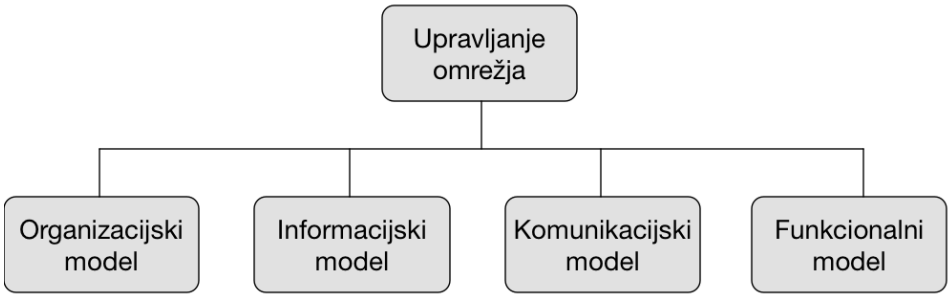
Upravljanje računalniških omrežij, naprav in storitev

V tem poglavju predstavimo kako pristopiti k upravljanju računalniških omrežij, naprav in storitev. Najprej v razdelku 2.1 predstavimo formalno definicijo upravljanja računalniških omrežij, ki je zapisana v modelu ISO/OSI [45]. S sestavnimi deli definicje se namreč sreča vsak, ki načrtuje arhitekturo sistema za upravljanje računalniških omrežij.

Nato v razdelku 2.2 predstavimo aktualne pristope k upravljanju in pomen programskega upravljanja omrežij. Preverili smo, ali je za programsko upravljanje omrežja nujno ločevanje nadzorne in podatkovne ravnine ali obstaja tudi drug pristop k upravljanju, ki je izvedenka popularnih programsko opredeljenih omrežij.

Po izbiri pristopa za upravljanje je nujna odločitev, kakšna bo oblika upravljalske informacije, ki si jo bo upravljalski sistem izmenjeval z omrežnimi napravami, da jo bo razumel najširši nabor omrežnih naprav. V razdelku 2.3 zato raziščemo možnosti za predstavitev nastavitvenih podatkov in podatkov o operativnem stanju naprav in statistiki njihovega delovanja.

Nadalje v razdelku 2.4 predstavimo, kateri protokoli za upravljanje računalniških omrežij so na voljo in jih ovrednotimo z namenom ocene njihove primernosti za uporabo pri programskem upravljanju omrežij.



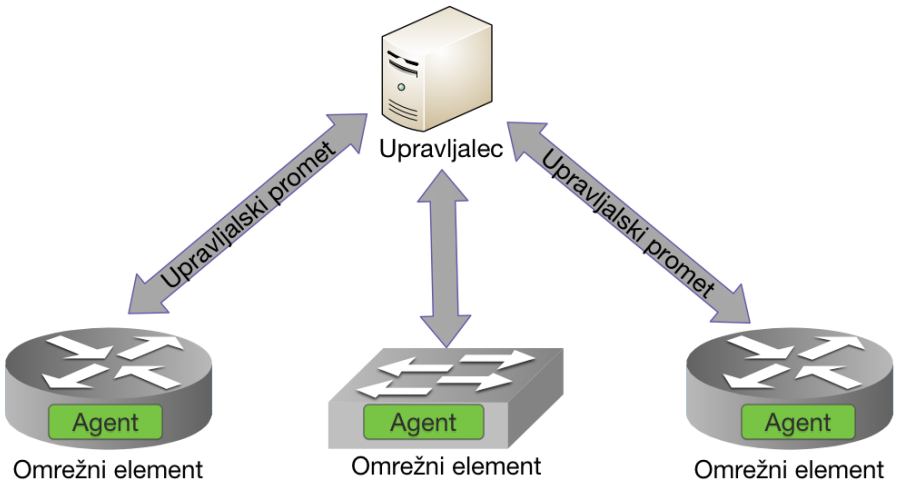
Slika 2.1: ISO/OSI model za upravljanje omrežij [45].

V razdelku 2.5 predstavimo dela strokovnjakov, ki so ocenjevali nove protokole za upravljanje z namenom uporabe v sodobnih pristopih k upravljanju in dela, v katerih so razvijali sisteme za upravljanje, s katerimi bi nadomestili tradicionalno upravljanje računalniških omrežij. Kot iztočnico v razdelku 2.6 zapišemo odprta vprašanja, ki jih skupnost še rešuje.

2.1 Model upravljanja omrežij

Model upravljanja omrežij ISO/OSI upravljanje omrežij deli na štiri dele, ki so prikazani na sliki 2.1. To so:

1. **Organizacijski model (*ang. Organization model*)**: Opisuje komponente upravljanja računalniških omrežij. Ta so sestavljena iz *omrežnih elementov*, kot so stikala, usmerjevalniki, tiskalniki, telefoni IP ipd. Omrežne elemente lahko delimo v dve skupini; tiste, ki jih upravljamo, in tiste, ki jih ne. Omrežni elementi, ki jih upravljamo, imajo običajno programsko opremo, imenovano *agent (ang. Agent)*. Agent protokola deluje na upravljani napravi in shranjuje informacije o delovanju naprave in njenih nastavitvah. Odgovarja na zahteve upravljalca in generira obvestila, da upravljalca obvešča o določenih dogodkih. *Upravljalec (ang. Manager)* je aplikacija, ki nadzoruje in upravlja naprave.



Slika 2.2: Upravljalec je aplikacija, ki upravlja enega ali več omrežnih elementov. Z omrežnimi elementi komunicira s pomočjo agentov, ki delujejo na posameznem upravljanem elementu.

To počne tako, da periodično pošilja zahtevke agentom, ki delujejo na napravah. Slika 2.2 prikazuje organizacijski model upravljanja omrežij.

2. **Informacijski model (*ang. Information model*)**: Ukvarja se s strukturo in shranjevanjem podatkov, ki si jih izmenjujeta upravljalec in agent. Skrbi tudi za predstavitev upravljalskih objektov in povezavo med njimi.

3. **Komunikacijski model (*ang. Communication model*)**: Definira, kako se podatki prenesejo med upravljalcem in agentom ali med upravljalskimi procesi. Pokriva tri področja komunikacije med dvema entitetama, *transportni medij* med entitetami, *obliko sporočil* in *operacije*, ki so dejanski ukazi in odzivi nanje.

4. **Funkcijski model (*ang. Functional model*)**: Model naslavlja funkcionalne komponente aplikacij, ki jih potrebujemo za učinkovito in sistematično upravljanje računalniških omrežij. Sestavljen je iz pe-

tih skupin, to so *upravljanje nastavitev (ang. Configuration management)*, *odkrivanje napak (ang. Fault management)*, *optimizacija delovanja (ang. Performance management)*, *varnost (ang. Security management)* in *beleženje (ang. Accounting)*.

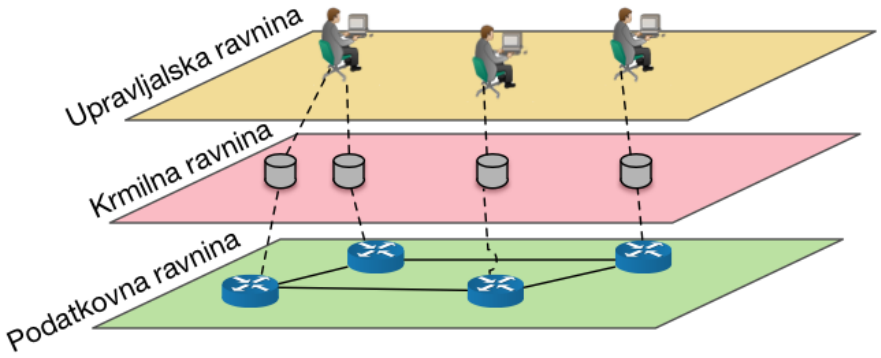
2.2 Pristopi k upravljanju

Pristop k gradnji in upravljanju omrežij se je skozi čas zaradi zahtev uporabnikov spreminjal. Z novimi pristopi h gradnji so se razvili tudi novi pristopi k upravljanju, ki zamenjujejo tradicionalni pristop, ki zaradi potrebe po večji dinamičnosti in razširljivosti omrežij ni bil več primeren. Najprej je kot izhodišče v razdelku 2.2.1 predstavljen tradicionalni pristop in nato v razdeku 2.2.2 pristop, ki je bil v zadnjih letih v skupnosti in industriji deležen največ pozornosti. To so programsko opredeljena omrežja. Programsko opredeljena omrežja na novo definirajo vloge pozameznih gradnikov v omrežju in obljubljajo izboljšave tudi na področju upravljanja. Velike spremembe omrežne arhitekture, ki jih prinašajo programsko opredeljena omrežja so se za določene primere uporabe izkazale kot neprimerne. To je spodbudilo razvoj novega pristopa k upravljanju, ki je predstavljen v razdelku 2.2.3 in se osredotoča predvsem na iskanje rešitev za avtomatizacijo nalog, s katerimi se vsakodnevno srečamo pri upravljanju omrežij.

Za lažje razumevanje pristopov h gradnji in k upravljanju omrežne infrastrukture je na sliki 2.3 prikazana delitev omrežja na tri ravnine - *upravljalsko*, *krmilno* in *podatkovno*. Podatkovna ravnina skrbi za učinkovito posredovanje omrežnih paketov. Krmilna je zadolžena za polnenje tabel, ki odločajo, kako in kam se paketi pošiljajo. Upravljalska ravnina je sestavljena iz programske opreme za upravljanje (npr. upravljalca protokola SNMP).

2.2.1 Tradicionalno upravljanje

Tradicionalno upravljanje se danes še vedno pogosto uporablja. Upravljanje izvaja človeški operater, ki z uporabo programske opreme upravlja enega

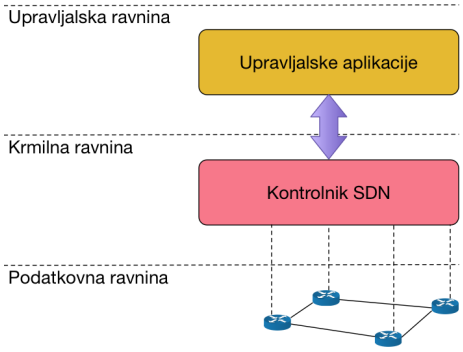


Slika 2.3: Računalniško omrežje, razdeljeno na ravnine.

ali več omrežnih elementov. V tradicionalnih omrežjih se krmilna in podatkovna ravnina izvajata na omrežnem elementu, upravljalska pa v programski opremi za upravljanje, s katero rokuje operater. Programska oprema je lahko program za dostop do ukazne vrstice, npr. TELNET, SSH ali program, ki deluje v vlogi upravljalca (npr. upravljalec SNMP) in z uporabo upravljalskega protokola upravlja omrežne elemente. Pri tem je pomembno, da pri tradicionalnem pristopu programska oprema običajno za upravljanje podpira le en upravljalski protokol. Prav tako je programska oprema za upravljanje pogosto lastniška in podpira le upravljanje omrežnih elementov enega proizvajalca [29].

Človeški operater, ki upravljanje izvaja mora biti dobro usposobljen. Upravljanje, ki je sicer zelo zamudno, izvaja na večih omrežnih elementih, kar v primeru opreme različnih proizvajalcev od njega zahteva dobro poznavanje različnih uporabniških vmesnikov, prav tako pa obstaja veliko možnosti za napake. Učenje novih uporabniških vmesnikov je zapleteno in dolgotrajno.

Za poenostavitev pogostih upravljalskih opravil si spretnejši omrežni operaterji ukaze zapišejo v obliki skripte, ki njihove naloge naredi ponovljive in



Slika 2.4: Delitev ravnin v programsko opredeljenih omrežjih.

jim proces nastavljanja nekoliko skrajša.

2.2.2 Upravljanje v programsko opredeljenih omrežjih

Programsko opredeljena omrežja ločujejo krmilno in podatkovno ravnino. Delitev je prikazana na sliki 2.4. Krmilna ravnina se izvaja na centraliziranem krmilniku SDN, podatkovna pa na omrežnih elementih. Upravljalska ravnina je del zunanjih omrežnih aplikacij. Omrežni elementi lahko postanejo preprostejši, saj se vsa logika izvaja na centraliziranem krmilniku SDN. Vsa interakcija z omrežno infrastrukturo poteka preko krmilnika. Aplikacije lahko za interakcijo z abstraktno predstavitvijo omrežja uporabljajo programabilne vmesnike API, ki jih ponuja krmilnik SDN. Krmilnik SDN za upravljanje prometnih tokov in nastavitev na omrežnih elementih uporablja protokole kot so NETCONF, OpenFlow in OVSDB.

Delitev v teoriji prinaša nekaj prednosti, kot so lažje nastavljanje omrežja kot celote in zmanjšanje možnosti za napake. Poenostavlja odziv na spremembe na omrežju in nudi enostavnejše dodajanje novih storitev v omrežje, saj je vsa logika centralizirana in za potrebe nove storitve ni potrebno spreminjati posameznega omrežnega elementa [29].

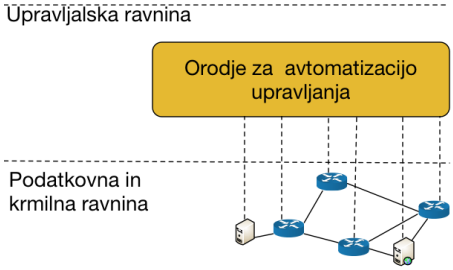
V praksi se vse bolj kažejo slabosti arhitekture programsko opredeljenih

omrežij. Avtorji v člankih [20, 29, 36] navajajo številne slabosti in kot eno glavnih izpostavljajo pomanjkanje standardizacije. Upravljalci, ki bi za poenostavitev upravljanja svoje omrežje zgradili po principu programsko opredeljenih omrežij, bi morali obstoječo omrežno opremo zamenjati s preprostejšo, ki bi izvajala le posredovanje podatkovnega prometa. Upravljanje omrežne infrastrukture s centraliziranim krmilnikom SDN, ki izvaja vse odločitve v omrežju pomeni, da krmilnik postane ozko grlo omrežja. Centralizacija vse logike tudi močno upočasni prometne tokove v omrežju. Uporaba podatkovno opredeljenih omrežij je tako omejena le na določene primere uporabe, največkrat se uporablja za upravljanje infrastrukture v podatkovnih centrih. Avtorji članka [23] kot slabost pri uvajanju krmilnikov SDN v upravljanje omrežne infrastrukture vidijo tudi njihovo zapletenost in zahtevnost za integracijo.

2.2.3 Avtomatizirano upravljanje

Zaradi težavnosti vpeljave programsko opredeljenih omrežij se je razvil nov pristop k upravljanju, ki se ukvarja predvsem z avtomatizacijo ponovljivih upravljalskih opravil in njihovo standardizacijo. Pri tem omrežnih ravnin ne deli tako kot programsko opredeljena omrežja, ampak krmilno in podatkovno plast prepušča omrežnim elementom. Delitev je prikazana na sliki 2.5. To pomeni, da za uvedbo pristopa ni potrebna menjava omrežne opreme. Upravljanje z orodji za avtomatizacijo se je na omrežno infrastrukturo preneslo s strežniške. To pomeni, da lahko poleg omrežnih naprav omogoča tudi upravljanje storitev na strežnikih, kar nudi dobro izhodišče za enovito upravljanje celotne infrastrukture in zmanjšuje število upravljalskih sistemov, za katere mora upravljalec skrbeti.

Trenutno izziv avtomatiziranega upravljanja predstavljajo številni upravljalski vmesniki zelo heterogenih omrežnih naprav. Za podporo omrežne opreme različnih upravljalcev morajo sistemi poznati številne protokole in načine za predstavitev upravljalske informacije. V skupnosti in industriji zato potekajo številne iniciative, ki želijo heterogenost upravljanja rešiti in



Slika 2.5: Delitev ravnin pri avtomatiziranem upravljanju.

zagotoviti standardizirano predstavitev upravljalskih informacij, ki jih bodo nekoč razumele vse omrežne naprave.

2.3 Predstavitev upravljalske informacije

S potrebo po enovitem upravljanju in željo, da se upravljalcem ne bo več potrebno za vsako omrežno napravo učiti novega upravljalskega vmesnika, je skupnost veliko truda vložila v enotno predstavitev upravljalske informacije. Enotna predstavitev je pomembna za zagotovitev nivoja abstrakcije, ki programski opremi za upravljanje zagotavlja enoteno obliko informacij, ki jih razumejo vse naprave.

V tem razdelku sta predstavljena dva jezika za predstavitev upravljalske informacije. Prvi se imenuje SMI (*ang. Structure of management information*) in je predstavljen v razdelku 2.3.1. Razvit je bil za potrebe upravljalskega protokola SNMP, ki je predstavljen v razdelku 2.4.5. Zaradi slabosti protokola SNMP in novih zahtev za upravljanje in nastavljanje omrežij je skupnost razvila nov protokol za upravljanje, NETCONF, ki je predstavljen v razdelku 2.4.6, in skupaj z njim tudi nov jezik za modeliranje upravljalskih informacij YANG, ki je opisan v razdelku 2.3.2. Oba jezika sta bila razvita za potrebe določenega protokola, a v splošnem njuna uporaba ni vezana neposredno na protokol.

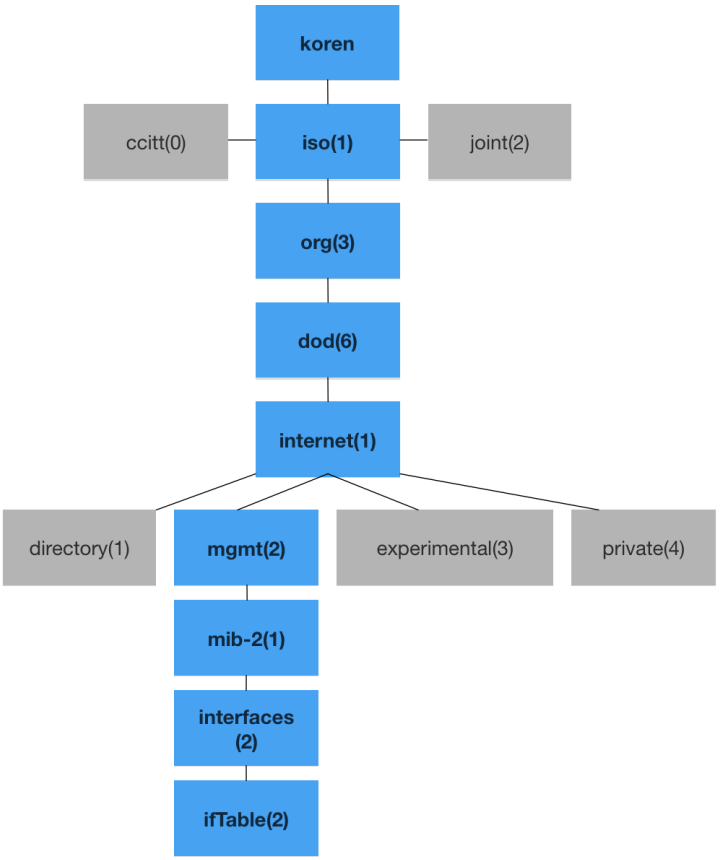
2.3.1 SMI

SMI (*ang. Structure of Management Information*) [39] definira pravila za predstavitev upravljalske informacije. Posamezno enoto informacije poimenuje upravljalski objekt. SMI je bil razvit za potrebe definicije upravljalskih informacij v protokolu za upravljanje SNMP. Upravljalski objekti [45] predstavljajo podatke o nastavitvah ali podatke o operativnem stanju omrežnega elementa. SMI določa, da mora imeti vsak upravljalski objekt določeno ime, sintakso in kodiranje. Ime objekta je določeno z enoličnim identifikatorjem (*Object ID - OID*). Imena so lahko berljiva človeku ali numerična. Poimenovanje temelji na imenski shemi ASN.1, ki omogoča hierarhičnost objektov. Ime objekta je zaporedje števil v drevesu, ki so ločeni s piko. Drevo upravljalskih objektov se v protokolu SNMP imenuje drevo MIB. MIB (*ang. Management Information Base*) je baza s podatki opisov vseh upravljalskih objektov na omrežnem elementu. V primeru na sliki 2.6 je objekt *ifTable* enolično identificiran z *.1.3.6.1.2.1.2.2* oziroma v tekstovni obliki *iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable*.

Za opis sintakse in tipov SMI uporablja podmnožico standarda ASN.1 (*ang. Abstract Syntax Notation 1*). Instanca upravljalskega objekta mora biti za pošiljanje po omrežju pretvorjena v obliko, ki jo določa BER (*ang. Basic Encoding Rules*). BER je format za kodiranje sporočil, ki se pošiljajo med agentom in upravljalcem. Format pretvori ASCII sporočila v binarno obliko. Da to doseže uporablja princip tip-dolžina-podatek (*ang. type-length-value- TLV*) V vsakem sporočilu se tako pošlje tip podatka, dolžina podatkov in dejanski podatek.

SMI definira tudi podatkovne tipe, ki so lahko preprosti ali sestavljeni. Preprosti so ekvivalenti primitivom v programskih jezikih (integer itd.). Sestavljeni tipi so tipa OBJECT-TYPE in MODULE-TYPE.

Uporabniki jezika SMI so izpostavili njegove slabosti [41]. Označili so ga za zelo težkega za uporabo in nepraktičnega. Izpostavili so pomanjkanje strukturiranih tipov in možnosti modeliranja komunikacije RPC (*ang. remote procedure call*). V upravljalskih objektih, definiranih z uporabo je-



Slika 2.6: Prikaz drevesne strukture upravljalske informacije, ki jo definira SMI.

zika SMI, podatki o nastavitvah omrežnih elementov in izmerjeni podatki o njenem stanju naprave niso ločeni. Posledica zapletene uporabe je tudi, da proizvajalci upravljalske objekte za nove funkcionalnosti omrežnih elementov dodajajo prepočasi. Za uporabo nove funkcionalnosti se zato namesto protokola SNMP uporabljajo drugi protokoli.

2.3.2 YANG

YANG [16] je jezik, ki je bil razvit za modeliranje upravljalskih informacij v protokolu NETCONF, ki je opisan v razdelku 2.4.6. Nastal je na temeljih neuspelega projekta za prenovo jezika za modele SMIng [42], ki je želel rešiti težave z uporabo jezika SMI. YANG je uporabljen tudi za definicijo klicev oddaljenih procedur protokola in obvestil v protokolu NETCONF.

Jezik je zasnovan tako, da je lažje berljiv [19]. Upravljalske informacije definirane z jezikom YANG, so vsebovane v modelih, ki so identificirani z imenskim prostorom XML. Jezik je razširljiv in omogoča dodajanje novih modelov ali razširjanje obstoječih z dodatnimi podatki (*ang. augment*). Prav tako omogoča deljenje modulov na podmodule. Razširitve lahko definirajo organizacije za standardizacijo, proizvajalci ali posamezniki.

Jezik ima nekaj osnovnih podatkovnih tipov in nabor tipov, ki se pogosto uporabljajo v omrežjih (naslov IP, števec, itd.). Vsebuje gradnike za razvoj dodatnih podatkovnih tipov. YANG dovoljuje definicijo skupin elementov, ki jih nato lahko ponovno uporabimo ali razširimo z uporabo gradnika *grouping*.

Za predstavitev podatkov YANG uporablja XML. Vse podatke shranjuje v listih drevesa XML. Vsak list predstavlja eno vrednost. Jezik podpira definicijo zapletenih struktur, ki imajo lahko več instanc v obliki seznamov. Jezik strogo ločuje nastavitvene podatke in podatke o stanju omrežnega elementa. Nastavitveni podatki so označeni z značko *config*. Instance modelov YANG so kodirane v obliki XML.

Od prve verzije so se pojavili predlogi, da se ga uporabi še za modeliranje upravljalskih informacij pri drugih protokolih [17], s katerimi bi lahko nastavljali naprave, to sta protokola RESTCONF [14] in CoMI [15]. Z uporabo

za predstavitev upravljalskih informacij drugih protokolov, se jezik nekoliko ločuje od protokola NETCONF. Predloga se trenutno v obliki osnutka pripravljata v IETF. Prav tako so v postopku razvoja tudi druge oblike za kodiranje klicev procedur, ki bi obstoječemu kodiranju XML dodale še format JSON [31], ki se pogosto uporablja v spletnih aplikacijah in pri arhitekturi REST.

Trenutno največji izziv za skupnost predstavlja standardizacija modelov YANG, ki jih bodo razumele naprave širokega nabora proizvajalcev. Odprto vprašanje je podrobno predstavljeno v razdelku 2.6.

2.4 Protokoli za upravljanje

Z namenom ugotoviti, kako primeren je posamezen upravljalski protokol za uporabo v sodobnih pristopih k upravljanju *(gl. razdelek 2.2)*, jih v tem razdelku predstavimo in ovrednotimo.

Kot izhodišče za ovrednotenje protkolov v razdelku 2.4.1 predstavimo pregled zahtev za bodoče upravljalske protokole, ki so jih že leta 2002 na delavnici v okviru IAB (*Internet Architecture Board*) zapisali upravljalci računalniških omrežij. Nato v razdelku 2.4.2 predstavimo dokument, ki definira kazalnike, s katerimi lahko ocenimo uspešnost protokolov. V razdelku 2.4.3 na podlagi zahtev v dokumentu in raziskave področja določimo sloje, s katerimi si bomo pomagali pri predstavitvi in ovrednotenju protokolov.

V razdelkih od 2.4.4 do 2.4.7 pregledamo aktualne protokole za upravljanje in njihove lastnosti. Predstavljene protokole v razdelku 2.4.8 ovrednotimo. Kazalnike za ovrednotenje smo določili s pomočjo zahtev z delavnice, s pomočjo kazalnikov v dokumentu za oceno uspešnosti protokolov [8] in s kazalniki, ki smo jih določili med raziskavo področja.

2.4.1 Delavnica

Dokument [41] vsebuje pregled delavnice, ki jo je leta 2002 organiziral IAB. Namen delavnice je bil dialog med upravljalci računalniških omrežij in raz-

vijalci protokolov, da bodo delovne skupine lahko bolje načrtovale omrežne protokole prihodnosti.

Spodaj so našteta do takrat nerešena vprašanja, ki jih bodo morale reševati nove tehnologije za upravljanje računalniških omrežij.

Preprostost uporabe je pomembna za katerokoli tehnologijo za upravljanje omrežij.

Jasna ločitev podatkov o nastavitvah in o operativnem stanju, saj je trenutno na precej napravah težko ugotoviti, katere nastavitve so lahko nastavljene, katere pa so pridobljene iz drugih virov, npr. usmerjevalnih protokolov.

Ločeno pridobivanje podatkov mora nuditi možnost, da podatke o nastavitvah in o oprativnem stanju pridobimo ločeno. Omogočena mora biti tudi možnost primerjave podatkov med napravami.

Nastavljanje omrežja kot celote je zaradi večje dinamike pomembno, zato mora biti upravljalcu omogočeno upravljanje omrežja kot celote in ne le posamezne naprave.

Transakcije nastavitev na več naprav bi zelo poenostavile upravljanje.

Generiranje minimalnega seta sprememb mora, če sta podani konfiguracija A in konfiguracija B, generirati spremembe, potrebne, da pridemo iz stanja A v stanje B, tako da je izvedeno minimalno število posameznih sprememb.

Izvoz in uvoz nastavitev iz omrežne naprave morata biti omogočena. Zaželjeno je, da so ti mehanizmi standardizirani.

Preverjanje konsistentnosti nastavitev mora biti preprosto.

Skupni modeli za nastavitve, ki so običajno shranjene v bazi in nato pretvorjene v obliko, ki deluje na končni napravi.

Procesiranje nastavitvenih in operativnih podatkov mora biti preprosto. Podprta morajo biti orodja za primerjavo besedila, kot je diff, in orodja za upravljanje verzij dokumentov, kot sta git ali svn.

Nadzor dostopa mora biti granularen in mora omogočati nekaj pogosto uporabljenih načinov, kot je dodeljevanje pravic glede na vloge. Dostop po principu najmanj privilegijev, kar omogoča, da uporabnik dobi le toliko dostopa, kot ga potrebuje za izvedbo naloge.

Preverjanje konsistentnosti nastavitev med večimi napravami.

Hranjenje večih nastavitvenih datotek je zahteva, ki bi jo morale podpreti vse omrežne naprave. Pomembno je tudi, da lahko razlikujemo med aktivacjo določenih nastavitev in nastavitvijo določenega parametra.

Nadzor dostopa mora omogočati več tipov omejevanja dostopa, kot sta npr. *osredotočen na nalogo* ali *podatkovno osredotočen* dostop.

2.4.2 Kazalniki za oceno uspešnosti protokola

Dokument, ki vsebuje kazalnike za oceno uspešnosti protokola (*ang. What Makes for a Successful Protocol?*) [8] na podlagi raziskav definira pomembne kazalnike, ki vplivajo na uspeh ali neuspeh protokolov. Glavni namen je določiti, kako doseči, da bo določen protokol, definiran v IETF, čimbolje sprejet v skupnosti in industriji.

Nabor kazalnikov definiranih v dokumentu [8]:

- **reševanje potrebe** (*ang. Meet a Real Need*),
- **postopno vzpostavljanje** (*ang. Incremental Deployability*),

- **dostopnost odprtokodnih orodij in implementacije protokola** *ang. Open Code Availability*),
- **prosta uporaba** (*ang. Freedom from Usage Restrictions*),
- **odprta specifikacija** (*ang. Open Specification Availability*),
- **odprt vzdrževalni proces** (*ang. Open Maintenance Processes*),
- **dobra tehnična zasnova** (*ang. Good Technical Design*),
- **razširljivost** (*ang. Extensible*),
- **povečljivost** (*ang. No Hard Scalability Bound*),
- **zadovoljiva odpornost na varnostne grožnje** (*ang. Threats Sufficiently Mitigated*).

2.4.3 Sloji protokolov

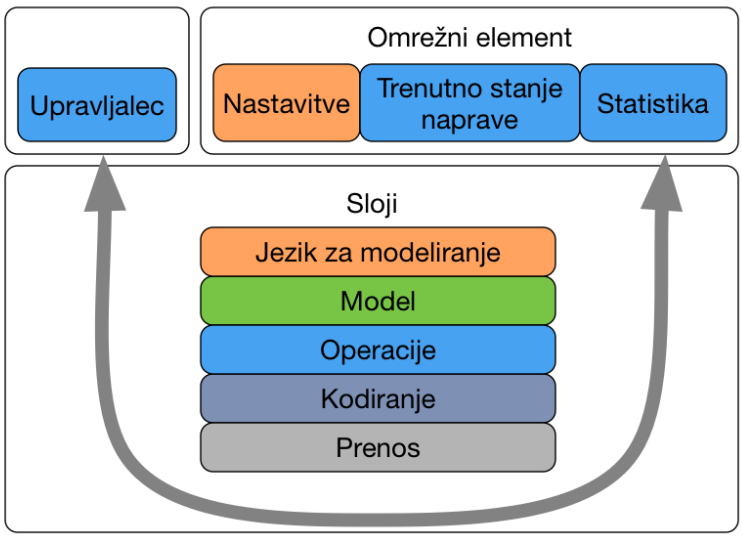
S sloji [13] upravljalskih protokolov, ki so prikazani na sliki 2.7, lahko prikažemo sestavo posameznega protokola in raziščemo, kako definira upravljalsko informacijo in njeno izmenjavo med upravljalcem in omrežnim elementom. Upravljalske informacije lahko predstavljajo *nastavitve*, *trenutno stanje* ali *statistiko* omrežnega elementa.

Jezik za modeliranje in model določata predstavitev upravljalske informacije. Možnosti za učinkovito modeliranje smo predstavili v razdelku 2.3.

Operacije definirajo akcije, s katerimi si upravljalec in omrežni element lahko izmenjata upravljalsko informacijo.

Kodiranje opisuje, kako in v kakšni obliki se zapiše upravljalska informacija, preden si jo upravljalec in omrežni element izmenjata.

Prenos definira podprte protokole, s katerimi se upravljalska informacija dostavi od izvora do ponora.



Slika 2.7: Sloji protokolov za upravljanje računalniških omrežij in storitev.

2.4.4 Ukazna vrstica

Ukazna vrstica (*ang. Command line interface - CLI*) je tekstovni vmesnik za upravljanje naprav in storitev. Upravljalec se v ukazno vrstico prijavi z uporabo sejnega protokola, kot sta SSH ali TELNET. Zaradi zagotavljanja varnosti in šifriranja je bolj priporočljiva uporaba protokola SSH, a se tudi TELNET pogosto uporablja. Upravljalec dobi neposreden dostop do upravljalskega nivoja naprav. Formalni jezik za modeliranje vmesnikov ne obstaja, zato se ti med proizvajalci ali celo med napravami istega proizvajalca lahko zelo razlikujejo. Slabost slednjega je, da se mora upravljalec zaradi običajne velike heterogenosti računalniških omrežij naučiti uporabljati veliko različnih vmesnikov. Ti vmesniki so namenjeni človeškim upravljalcem, a se pogosto uporabljajo za interakcijo s sistemi za upravljanje z omrežji (*ang. Network management system*). Sistemi za upravljanje z omrežji odzive tekstovnega vmesnika procesirajo in se nato ustrezno odzovejo. Pristop je skupnost poimenovala strganje podatkov z zaslona (*ang. Screen scraping*). Takšen sistem se lahko izkaže kot izjemno nepredvidljiv, saj se lahko odzivi tekstovnega vmesnika med nadgradnjami programske opreme spremenijo, zato je potrebno stalno prilagajanje sistema.

2.4.5 SNMP

SNMP [21] je protokol za upravljanje in nadzor na aplikacijskem nivoju ISO/OSI, ki ga je definiral IETF. Služi za izmenjavo upravljalskih sporočil med upravljalcem in nadzorovanimi napravami v omrežju. Za podporo upravljanju na omrežnem elementu teče programska oprema, imenovana agent.

Formalni jezik, ki definira strukturo, poimenovanje in vsebino upravljalske informacije, ki si jo izmenjujeta upravljalec in omrežni element, se imenuje standard SMI *(gl. razdelek 2.3.1)*. Baza, v kateri so shranjene izmerjene in nastavljene informacije na omrežnem elementu, se imenuje MDB (*ang. Management database*).

Protokol definira nabor operacij [45] oziroma sporočil, ki jih lahko upo-

rabljata upravljalec in agent.

- **GetRequest:** upravljalec agentu pošlje zahtevek za vrednost iz baze MIB. Agent na zahtevek odgovori s sporočilom tipa Response.
- **GetNextRequest:** upravljalec zahteva vrednost naslednjega objekta v drevesu MIB.
- **GetBulkRequest:** je optimizirana verzija sporočila GetNextRequest in omogoča, da agent upravljalcu pošlje večji blok podatkov.
- **SetRequest:** to sporočilo upravljalcu omogoča, da upravljani napravi nastavi ali spremeni vrednost izbranega objekta.
- **Response:** je odgovor na sporočila *GetRequest*, *SetRequest*, *GetNextRequest*, *GetBulkRequest*, ki ga upravljalec pošlje agentu. Lahko vsebuje tudi kodo napake.
- **Trap:** je asinhrono obvestilo o nekem dogodku, ki ga agent pošlje upravljalcu.

Za prenos informacij se uporablja protokol UDP. Protokol za pošiljanje in prejemanje zahtev uporablja vrata 161, za prejemanje obvestil pa vrata 162.

SNMP je bil od prve verzije precej posodobljen, zadnja verzija je SNMPv3. Predvsem je s posodobitvami napredoval pri zagotavljanju večjega višje ravni. Starejši verziji 1 in 2 sta imeli namreč varnost zagotovljeno le z uporabo imena skupnosti (*ang. community*). Skupnost je bila uporabljena za definiranje upravljalskih pravic dostopa. Ime skupnosti je bilo mogoče ugotoviti z uporabo prisluškovanja prometa na omrežju, saj komunikacija ni bila šifrirana. V verziji 3 je protokol pridobil dva nova varnostna modela, to sta USM (*ang. user-based security model*) in VACM (*ang. view-based access model*), ter šifriranje komunikacije med upravljalcem in omrežnim elementom.

Medtem ko je protokol SNMP podprt na širokem naboru omrežnih elementov, se objekti MIB, ki jih elementi podpirajo, močno razlikujejo [41].

Standardni objekti MIB so pogosto na voljo le za branje in jih ni mogoče spreminjati. Prav tako na omrežnega elementa običajno z uporabo protokola SNMP ne moremo v celoti nastaviti. Protokol je zato najbolj primeren le za bralni dostop, ki ga uporabljajo nadzorni sistemi.

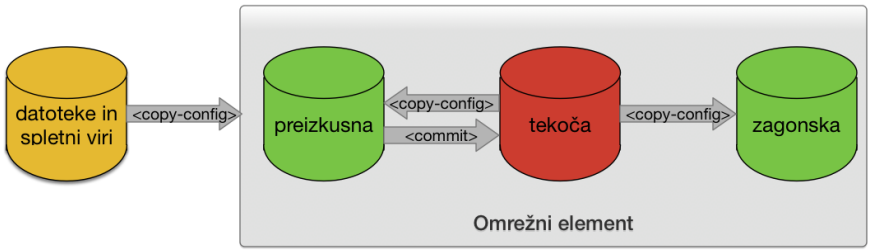
2.4.6 NETCONF

NETCONF [18] je osnovni gradnik za programsko upravljanje, nastavljanje računalniških naprav. Protokol omogoča napravam, da upravljalcu izpostavijo poln, formalen vmesnik (*ang. Application programming interface*), preko katerega lahko prejema polne ali delne nastavitvene podatke [47]. Podatke delimo v dva razreda, na nastavitvene in tiste, ki opisujejo stanje naprave. Protokol dovoljuje, da so funkcionalnosti upravljalskega protokola čim bližje lastnostim naprave, kar zmanjšuje čas za implementacijo novih funkcionalnosti, upravljalski sistemi pa dobijo neposreden dostop do vsebine v sintaksi in semantiki naprave. Komunikacija v protokolu temelji na klicih oddaljenih procedur (*ang. Remote procedure call - RPC*), za prenos se uporablja povezavno orientiran protokol. Obvezno mora biti podprt protokol SSH. Komunikacija poteka med upravljalcem, ki se imenuje odjemalec, in omrežnim elementom, ki se imenuje strežnik.

Oblika upravljalske informacije, ki si jo izmenjujeta odjemalec in strežnik, je modelirana z uporabo jezika YANG, ki je predstavljen v razdelku 2.3.2. YANG definira tudi klice oddaljenih procedur protokola in obvestila v protokolu NETCONF. Upravljalske informacije in klici procedur so kodirani v obliki XML (*ang. Extensible Markup Language*).

Protokol omogoča uporabo več baz za shranjevanje izmerjenih in nastavljenih podatkov. Baze so prikazane na sliki 2.8. Slika prikazuje štiri baze: [13].

- *datoteke in spletni viri (ang. files and web sources)*, ki je namenjena uvozu nastavitev iz spleta,
- *preizkusna (ang. candidate)*, ki je namenjena preizkušanju novih na-



Slika 2.8: Prikaz baz za nastavitve na omrežnih elementih.

stavitev,

- *tekoča (ang. running)*, ki vsebuje trenutno veljavne nastavitve,
- *zagonska (ang. startup)*, ki vsebuje zagonske nastavitve.

Podpora več baz za nastavitvene podatke omogoča napredne pristope k upravljanju. Primer naprednega pristopa je preizkus novih nastavitev pred uveljavitvijo, kar zagotavlja višjo raven zanesljivosti pri nastavljanju. Tega izvedemo tako, da nastavitve najprej zapišemo v preizkusno bazo, v kateri nastavitve še ne vplivajo na delovanje omrežnega elementa, nato pa jih v primeru uspešnega preikusa, kopiramo v tekočo bazo, kjer so shranjene nastavitve, ki so trenutno v uporabi.

Na začetku komunikacije si upravljalec, ki ga protokol imenuje odjemalec, in omrežni element, poimenovan strežnik, izmenjata zmogljivosti (*ang. capabilities*). Slednja odjemalcu pove, katere operacije strežnik, s katerim komunicira, podpira, in svoje delovanje temu prilagodi. Osnovne operacije, ki jih morajo podpirati strežniki, so definirane v YANG modelu z oznako *urn:ietf:params:netconf:base:1.1*. To so:

- **close-session:** zaprtje trenutne seje,
- **copy-config:** kopiraj bazo za nastavitve,

- **delete-config:** izbriši bazo za nastavitve,
- **edit-config:** posodobi nastavitve v bazi,
- **get:** pridobi trenutne ali statistiko,
- **get-config:** pridobi podatke iz baze s trenutno veljavnimi nastavitvami,
- **kill-session:** zapri neko drugo sejo,
- **lock:** zakleni bazo, da se lahko vanjo piše le iz trenutne seje,
- **unlock:** odkleni bazo, da lahko vanjo pišejo vsi.

Protokol izpolnjuje vse zahteve iz dokumenta, ki je predstavljen v razdelku 2.4.1. Trenutno je največja pomanjklivost protokola NETCONF, ta da proizvajalci za predstavitev upravljalske informacije pripravljajo lastniške modele YANG, ki med seboj niso združljivi, pogosto pa celo različne omrežne naprave istega proizvajalca podpirajo različne modele.

2.4.7 REST

REST (*ang. Representational State Transfer*) [24] je arhitekturni stil za razvoj spletnih aplikacij. Za komunikacijo uporablja protokol HTTP (*ang. HyperText Transfer Protocol*). Arhitektura določa lastnosti, ki jih mora aplikacija izpolnjevati, da jo lahko opišemo kot RESTfull aplikacijo [38]. Ena izmed njih je tudi ločevanje odjemalca in strežnika, ki je običajna tudi pri drugih metodah za upravljanje računalniških omrežij. Za izpostavljanje spletnih storitev se pogosto uporablja arhitektura REST, ta vire izpostavi in drugim aplikacijam omogoči dostop do njih z uporabo URI (*ang. Uniform Resource Identifier*). Operacije, ki jih podpira za izvajanje operacij nad viri, so standardne metode HTTP, npr. OPTIONS, GET, PUT, POST, DELETE. Podatke se običajno izmenjuje v obliki JSON ali XML. Pri upravljanju z računalniškimi omrežji in storitvami REST pogosto nadomešča druge metode

za nastavljanje, predvsem ukazno vrstico in SNMP, ki sta težja za integracijo z zunanjimi aplikacijami, ki želijo vplivati na nastavitve omrežja. Pogosto se za to uporablja namenska programska oprema, nameščena na strežniku, ki ji lahko aplikacije izdajajo ukaze, ta pa jih nato posreduje naprej končni napravi. Za komunikacijo med namensko programsko opremo in končno napravo se še vedno uporabljajo drugi protokoli, kot so tekstovne datoteke v sintaksi ukazne vrstice ali SNMP. Shema za modeliranje podatkov v primeru uporabe vmesnikov REST običajno ne obstaja, določi jo proizvajalec oziroma razvijalec storitve. Tako kot ostale spletne storitve se za opis virov uporabi WADL (*ang. Web Application Description Language*). V primeru, da se za izmenjavo podatkov uporablja XML, pa se ga lahko definira z uporabo shem XML.

2.4.8 Ovrednotenje

Protokole, ki so predstavljeni v razdelkih od 2.4.4 do 2.4.7, smo ovrednotili z namenom ocene uporabnosti v sistemih za programsko upravljanje omrežij.

Najprej smo s pomočjo slojev, definiranih v razdelku 2.4.3, primerjali sestavne dele posameznega protokola. Razslojitev posameznega protokola je prikazana na sliki 2.9.

Med raziskavo smo določili kazalnike, ki pomembno vplivajo na uporabnost protokola. Pri izboru kazalnikov za ovrednotenje protokolov smo si pomagali z dokumentom, ki predlaga kazalnike za oceno uspešnosti protokola [8], saj so mnogi med njimi zanimivi s stališča uporabe upravljalskih protokolov v sistemih za programsko upravljanje omrežja. Celoten nabor kazalnikov je naštet v razdelku 2.4.2. Med njimi smo izbrali kazalnike: `odprta specifikacija`, `dostopnost odprtokodnih orodij in implementacije protokola`, `razširljivost`, `odpornost na varnostne grožnje`. Podmnožica izbranih kazalnikov najbolje opisuje lastnosti, ki jih mora imeti protokol, da je primeren za uporabo v sodobnih sistemih za upravljanje računalniških omrežij.

Kazalnike, ki smo jih črpali iz dokumenta, smo dopolnili z dodatnimi

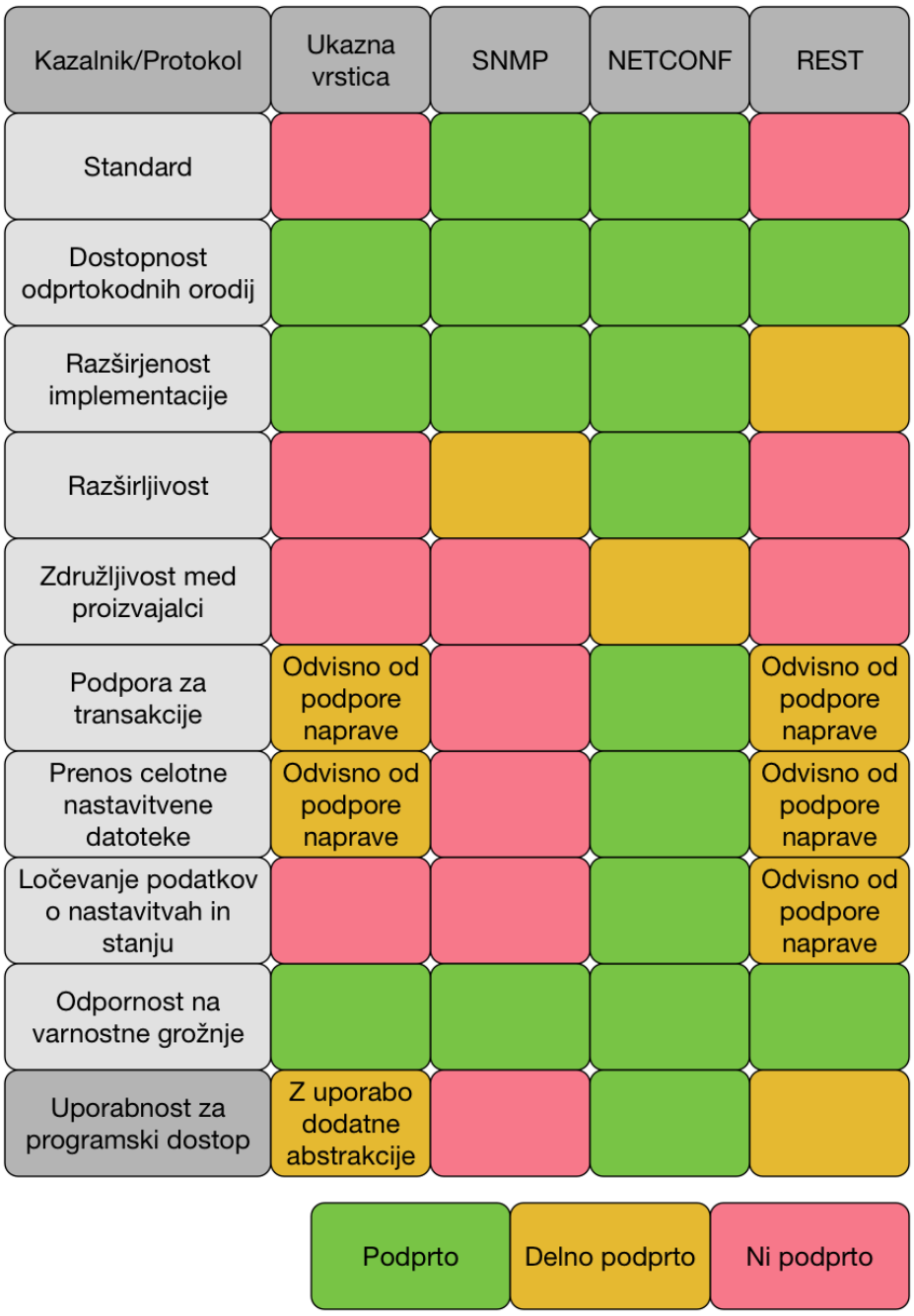
Sloj/Protokol	Ukazna vrstica	SNMP	NETCONF	REST
Jezik za modeliranje		SMI	Jezik YANG	XSD, WSDL
Model		MIB	Modeli YANG	
Operacije	Odvisno od podpore naprave	SNMP	NETCONF	Metode HTTP
Kodiranje	Tekst	BER	XML	XML, JSON
Prenos	SSH, TELNET TCP	UDP	SSH TCP	HTTPS TCP

Slika 2.9: Prikaz slojev v različnih protokolov za upravljanje naprav in storitev v računalniških omrežjih.

kazalniki, ki smo jih določili med raziskovanjem področja in se pogosto uporabljajo v skupnosti. Povzetek ovrednotenja je prikazan na sliki 2.10, podrobnejši opis kazalnikov in ovrednotenje protokolov predstavljajo naslednji odstavki.

Standardiziranost in odprta specifikacija nam pove, ali je metoda standardizirana s strani standardizacijskega telesa, kot je npr. IETF, odprtost specifikacije pa nam pove, če je specifikacija na voljo vsem, ki jo želijo uporabiti.

1. **Ukazna vrstica** in njeno delovanje nista standardizirana, zato se lahko interakcija z omrežnimi elementi med proizvajalci močno razlikuje, prav tako na voljo ni nobene odprte specifikacije, ki bi njeno delovanje definirala.



Slika 2.10: Primerjava protokolov na podlagi izbranih kazalnikov.

2. **SNMP:** Je standardiziran s strani IETF, specifikacija je dostopna vsem.

3. **NETCONF:** Je standardiziran s strani IETF, specifikacija je dostopna vsem.

4. **REST:** Ni standardiziran s strani nobenega standardizacijskega telesa in ni protokol, ampak zgolj arhitekturni stil za načrtovanje omrežnih spletnih aplikacij, ki ga je definiral Roy Thomas Fielding [24]. Pomanjkanje standardizacije zahteva velik programerski vložek za vključitev podpore za upravljanje posameznega omrežnega elementa v enovito infrastrukturo za upravljanje. Prav tako se lahko posamezni vmesniki REST močno razlikujejo med seboj.

Dostopnost odprtokodnih orodij in knjižnic za interakcijo s protokolom močno poveča uspešnost protokola in poenostavi njeno vključitev v sistem za upravljanje in nastavljanje omrežne infrastrukture.

1. **Ukazna vrstica:** Dostopnost je ustrezna, saj se za komunikacijo z ukazno vrstico omrežnih elementov pogosto uporablja protokola SSH ali TELNET, ki sta na različnih operacijskih sistemih dobro podprta. Za oba protokola obstajajo tudi knjižnice za programske jezike, v Pythonu se za vzpostavitev komunikacije SSH in izvajanje ukazov uporablja knjižnica *paramiko*.

2. **SNMP:** Za protokol obstaja velik nabor odprtokodnih orodij, najbolj poznana je zbirka aplikacij z oznako NET-SNMP [40], ki vključuje programe za delo v ukazni vrstici, grafični brskalnik za bazo MIB, vmesnik za sprejemanje obvestil SNMP in agenta za implementacijo na omrežnih elementih.

3. **NETCONF:** Za podporo protokola NETCONF in jezika za modeliranje nastavitev in operativnega stanja naprave YANG se je v industriji in odprtokodni skupnosti izdelalo veliko zbirko različnih orodij. Primer

knjižnice za programski jezik Python je *ncclient*, interakcijo z modeli YANG pa omogoča knjižnica *pyang*. Poleg tega obstajajo še sistemske knjižnice, ki so namenjene implementaciji odjemalcev ali strežnikov protokola NETCONF, npr. *libnetconf* in *libyang*.

4. **REST:** Ker je arhitektura REST pogosto uporabljena pri implementacijah spletnih storitev, zanjo obstaja veliko različnih knjižnic in orodij v praktično vseh programskih jezikih. Primeri takšnih so orodja *curl* in knjižnice za programski jezih python - *requests*. Prav tako obstaja širok izbor knjižnic za interakcijo z najpogostejšima oblikama za izmenjavo podatkov v arhitekturi REST, to sta XML in JSON. Primera knjižnic v programskem jeziku python sta *etree* in *json*.

Razširjenost implementacije na omrežnih elementih različnih proizvajalcev močno poveča uporabnost protokola.

1. **Ukazna vrstica:** Je zelo razširjena metoda in podprta praktično na vseh omrežnih elementih.

2. **SNMP:** Je dobro zastopana na omrežnih elementih, a pogosto omogoča le branje podatkov o nastavitvah in stanju naprave njihovo spreminjanje pa z uporabo objektov SNMP ni mogoče.

3. **NETCONF:** Implementiran na širokem naboru omrežnih elementov različnih proizvajalcev v industriji, a se implementacije močno razlikujejo. Do razlik prihaja predvsem v podpori različnih zmogljivosti, ki so definirane v obliki modelov YANG.

4. **REST:** Upravljanje z uporabo vmesnikov REST običajno podpirajo predvsem omrežni elementi, ki so namenjeni podatkovnim centrom in večji omrežni krmilniki, ki so namenjeni nadzoru gruče brezžičnih točk ali stikal.

Razširljivost protokolu omogoča, da se uporablja še za dodatne funkcionalnosti, ki v prvotni specifikaciji niso bile definirane. Pri *ukazni vrstici* in arhitekturi *REST* o razširljivosti težko govorimo, saj protokola nista standardizirana in obseg funkcionalnosti, ki jo posamezen proizvajalec implementiran, ni nikjer določen.

1. **SNMP:** Posebne razširljivosti ne omogoča, razen definiranja lastnih objektov MIB, ki jih lahko določi posamezen proizvajalec. Večjo prilagodljivost lahko dobimo le, če se standardu ne sledi v celoti.

2. **NETCONF:** Omogoča, da se osnovne module razširi. To lahko dosežemo z uporabo gradnika *augment* in definiranja dodatnih podatkovnih tipov.

Združljivost med proizvajalci je ključnega pomena za upravljanje, a če je metoda standardizirana, še ne pomeni, da bo komunikacija med omrežnimi elementi in upravljalcem različnih proizvajalcev izgledala enako. *Ukazna vrstica* in *vmesniki REST* različnih proizvajalcev običajno nimajo nič skupnega.

1. **SNMP:** Protokol definira osnovne objekte MIB, ki se uporabljajo za izmenjavo podatkov med upravljalcem in agentom, a v praksi proizvajalci omrežne opreme pogosto definirajo svoje MIB, ki bodisi omogočajo pridobivanje dodatnih podatkov iz omrežnega elementa bodisi le nadomeščajo tiste, ki so definirani v standardu. Različno definirani objekti MIB pomenijo, da moramo za podporo nove naprave vsakič dodati podporo in pridobiti MIB-e, ki jih podpira.

2. **NETCONF:** Protokol definira osnovne modele YANG, ki jih morajo naprave podpreti. Prav tako se IETF trudi specificirati modele za osnovne funkcionalnosti omrežnih elementov. Kljub vsemu je trenutno stanje podobno protokolu SNMP, kjer vsak proizvajalec razširi obstoječe modele in na napravah podpre le svoje. Z veliko podporo velikih

podjetij iz industrije, kot so Microsoft, Google, Yahoo, Facebook se v delovni skupini OpenConfig pripravlja modele, ki bi zadostili potrebam upravljalcev velikih omrežij. V sredini leta 2016 sta dva izmed največjih proizvjalcev omrežne opreme napovedala podporo za skupne modele YANG - OpenConfig.

Podpora za transakcije je pri izvajanju nastavljanja pomembna. Transakcija je nabor sprememb, ki omrežni element spravijo iz trenutnega v želeno stanje. Lastnosti transakcije omogočajo, da se spremembe izvedejo v celoti, sicer se sploh ne. Podpora transakcijam omogoča, da se upravljalcu ni potrebno zavedati, kakšno mora biti zaporedje ukazov, ki jih želi izvesti. Z uporabo transakcij se več sprememb lahko izvaja paralelno in ob tem ne pride do težav. Ko se spremembe uveljavijo, morajo trajno ostati v omrežnem elementu.

1. **Ukazna vrstica:** Nima podpore za *transakcije*, saj je pri izvajanju ukazov njihovo zaporedje pomembno. Prav tako je odvisno od proizvajalca, ali podpira *atomarnost* sprememb. Zaporedje ukazov mora spreminjati stanje v bazi za preizkus nastavitev in se uveljaviti šele, ko so bili vnešeni vsi. Takšen način podpira operacijski sistem za omrežne naprave *JunOS* proizvajalca Juniper. Druga pogosto implementirana možnost je, da posamezni ukazi neposredno vplivajo na trenutno veljavne nastavitve naprave, kar pa ne izpolnjuje zahteve po atomarnosti sprememb. Vrstni red vnosa ukazov je pomemben, prav tako lahko z uporabo več upravljalskih sej, ki istočasno izvajajo ukaze na omrežnem elementu, pride do težav, saj ni nujno, da bo naprava končala v želenem stanju. Tako nista izpolnjeni niti *izolacija* niti *konsistentnost*, je pa izpolnjena *trajnost*, saj se spremembe po uspešni izvedbi shranijo.

2. **SNMP:** Ne podpira transakcij, saj je pogosto potrebno eno spremenljivko objekta MIB nastaviti pred drugo, tega pa se mora zavedati upravljalec. Prav tako mora v primeru napake pri izvajanju zaporedja

več sprememb upravljalec omrežni element povrniti v prvotno stanje, v kolikor ena izmed sprememb ni bila uspešno izvedena.

3. **NETCONF:** Transakcije podpira že v protokolu in z zmogljivostjo *:confirmed-commit* omogoča tudi izvedbo transakcij nad več omrežnimi elementi. V primeru, da transakcija na enem omrežnem elementu ni uspela, se vsi omrežni elementi po določenem časovnem intervalu, v katerem so čakali na potrditev, vrnejo v stanje pred transakcijo.

4. **REST:** Podobno kot ukazna vrstica tudi vmesniki REST ne podpirajo transakcij, saj je tudi pri njih vrstni red izvajanja ukazov pomemben. Prav tako mora upravljalec v primeru, da en izmed ukazov spodleti, sam implementirati ustrezne mehanizme, da omrežni element vrne v stanje pred transakcijo.

Prenos celotne nastavitvene datoteke iz in v omrežni element je zelo pomembna lastnost. Omogoča, da nastavitve varnostno kopiramo z naprave in jih nato uvozimo nazaj na napravo. Lastnost je pomembna tudi zato, da lahko poiščemo razlike med nastavitvenimi datotekami z uporabo orodij, kot je *diff*.

1. **Ukazna vrstica:** Običajno omogoča izpis celotne nastavitvene datoteke na zaslon in njen prenos in izvoz z uporabo protokolov, kot so *scp*, *tftp*, *ftp*, *http*.

2. **SNMP:** Omogoča izpis vseh podatkov definiranih v bazi MIB, a ni jasne ločnice med nastavitvenimi podatki in podatki o operativnem stanju omrežnega elementa. Uvoz pridobljenih podatkov neposredno nazaj v omrežni element z namenom uvoza iz varnostnega kopiranja tako ni mogoč.

3. **NETCONF:** V protokolu je definirana operacija <*get-config* >, ki vrne celotne nastavitve shranjene za navedeno bazo za nastavitve.

4. **REST:** V večini implementacij vsak vmesnik REST spreminja le nastavitve specifičnega podsistema na omrežnem elementu, zato neposreden izvoz celotnih nastavitev naprave ni mogoč. Približek izvozu celotne datoteke bi bil izvoz in shranjevanje stanja s klicem vsakega vmesnika posebej in nato ob uvozu iz varnostne kopije ponovno klic vsakega vmesnika posebej. Nekateri proizvajalci podpirajo izpis celotnih nastavitev z uporabo vmesnikov REST.

Ločevanje nastavitvenih podatkov in podatkov o operativnem stanju naprave omogoča lažji in preglednejši dostop do podatkov, zato je priporočljivo, da metoda ločeno definira operativne podatke o omrežnem elementu in nastavitvene podatke. Pomemben je predvsem zato, da se programski opremi za upravljanje ni potrebno ukvarjati z ločevanjem, kaj določen podatek predstavlja. Pomembna lastnost je tudi, da so pridobljeni podatki v strukturirani obliki, npr. XML ali JSON.

1. **Ukazna vrstica:** Strogega ločevanja podatkov o operativnem stanju in nastavitvah omrežnega elementa običajno ni, npr. izpis podatkov o omrežnem vmesniku izpiše dele njegovih nastavitev in tudi podatke o števcih, napakah ipd.

2. **SNMP:** Objekti MIB brez posebne logične ločitve vsebujejo tako nastavitvene podatke kot tudi podatke o operativnem stanju vmesnika. Njihovo ločevanje na podlagi tipa zato ni mogoče.

3. **NETCONF:** Protokol strogo definira ločevanje operativnih podatkov o stanju naprave in nastavitvenih podatkih.

4. **REST:** Od implementacije vmesnikov je odvisno, ali lahko podatke o nastavitvah in stanju pridobimo ločeno ali preko istega klica REST. Pogosto vmesniki REST omogočajo le nastavljanje omrežnih elementov in je za dostop do podatkov o operativnem stanju omrežnega elementa potrebno uporabiti drugo metodo, npr. SNMP.

Odpornost na varnostne grožnje Uspešni protokoli običajno postanejo tarča napadov, zato je pomembno, da so zadosti zavarovani, da ne omogočajo zlorab. Na uspešnost vpliva tudi zadostna razširljivost, da se lahko potencialne varnostne luknje hitro odpravi.

1. **Ukazna vrstica:** Starejše implementacije so za komunikacijo med upravljalcem in omrežnim elementom pogosto uporabljale protokol TELNET, ki ni bil šifriran, danes pa se najpogosteje uporablja protokol SSH, ki zagotavlja varen komunikacijski kanal.

2. **SNMP:** Varnostne izboljšave je prinesla različica *SNMPv3*, ki je uvedla šifriran kanal med upravljalcem in agentom, ter nova varnostna modela USM (*ang. user-based security model*) in VACM (*ang. view-based access model*) [40].

3. **NETCONF:** Za komunikacijo uporablja protokol SSH, ki zagotavlja šifriranje podatkovnega kanala in overjanje odjemalca in strežnika z uporabo gesla in certifikatov.

4. **REST:** Varnost zagotavlja z uporabo protokola HTTPS za šifriranje podatkovnega kanala. Za overjanje odjemalca lahko uporablja različne metode, ki jih podpira protokol HTTP, kot so osnovna avtentikacija HTTP (*Basic HTTP authentication*), avtentikacija z žetoni, uporaba protokola OAuth.

Uporabnost za programski dostop nam pove primernost protokola za programsko upravljanje omrežnega elementa.

1. **Ukazna vrstica:** Ti vmesniki so namenjeni človeškim upravljalcem, a se pogosto uporabljajo tudi za interakcijo s sistemi za upravljanje z omrežji (*ang. Network management system*). Ti odzive tekstovnega vmesnika procesirajo in se nato ustrezno odzovejo. Pristopa se je v skupnosti prijel naziv strganje podatkov z zaslona (*ang. Screen scraping*). Takšen sistem se lahko izkaže kot izjemno nepredvidljiv,

saj se lahko odzivi tekstovnega vmesnika med nadgradnjami programske opreme spremenijo, zato je potrebno stalno prilagajanje sistema. Knjižnica NAPALM [10] kot eno izmed metod za interakcijo z napravami uporablja ukazno vrstico in interakcijo z ukazno vrstico abstrahira, da se programski opremi, ki jo uporablja, ni potrebno ukvarjati z odzivom ukazne vrstice. Uporaba knjižnic, kot je NAPALM, naredi uporabo ukazne vrstice pogojno uporabno, dokler omrežne naprave ne bodo podpirale primernejših protokolov.

2. **SNMP:** Je primarna izbira v mnogih rešitvah za zajemanje podatkov o operativnem stanju omrežnih naprav in storitev, ki se uporabljajo za nadzor omrežij. Za spreminjanje nastavitev pa ni najboljša izbira, saj se baze MIB med različnimi proizvajalci močno razlikujejo, prav tako ima veliko objektov MIB definiran le bralni dostop in jih ni mogoče spreminjati.

3. **NETCONF:** Zahteve in mehanizmi implementirani, v protokolu NETCONF, so najprimernejši za programabilni dostop. Prav tako uporabnost izboljšujejo jezik za modeliranje YANG in skupni modeli, specificirani v delovnih skupinah IETF in OpenConfig, ki bodo izboljšali združjivost med proizvajalci omrežne opreme. Koncepti so razširljivi še na prihajajoče protokole, kot je RESTCONF [14], ali uporabo v ogrodjih za avtomatizacijo, ki so opisana v razdelku 3.

4. **REST:** Je primerna izbira, če želimo omrežne elemente neposredno krmiliti iz aplikacije. Arhitekturni stil REST je namreč danes uporabljen za komunikacijo med zalednimi sistemi in uporabniškim vmesnikom v spletnih in mobilnih aplikacijah. Pri tem je potrebno poudariti, da sta implementacija in shema vmesnikov povsem prepuščeni posameznemu proizvajalcu, zato se lahko vmesniki med seboj močno razlikujejo. Podobno kot pri ukazni vrstici je za dodatek podpore za posamezno omrežno napravo potrebno razviti nov gonilnik, ki omogoča komunikacijo z njo. Prednost pred ukazno vrstico predstavljajo izhodni

statusi metod HTTP, ki omogočajo zaznavanje, ali je želen ukaz uspel ali ne.

Na podlagi ovrednotenja protokolov, ki je prikazano v tabeli na sliki 2.10, ocenjujemo, da je najprimernejši protokol za programsko upravljanje omrežij NETCONF, za predstavitev upravljalske informacije pa je najprimernejši jezik YANG. Ker NETCONF in YANG nista podprta na vseh omrežnih elementih, bomo v okviru pregleda obstoječih programskih rešitev, v naslednjem poglavju preverili, če je mogoče preostale protokole narediti primernejše za uporabo v sodobnih pristopih za upravljanje omrežij in kako to doseči.

2.5 Sorodna dela

Z rastjo kompleksnosti in velikosti računalniških omrežij je njihovo upravljanje postalo pomembnejše. V pričujočem razdelku so opisana aktualna dela strokovnjakov in raziskovalcev. V prvem članku [9] avtorji ovrednotijo nov protokol za upravljanje. V preostalih avtorji z namenom izboljšanja trenutnega pristopa k upravljanju svoje omrežne infrastrukture opišejo razvoj lastne rešitve ali ovrednotenje in izbiro rešitve iz nabora obstoječih rešitev.

Avtorji v članku [9] predstavljajo študijo protokola NETCONF [18]. Protokol primerjajo z obstoječimi protokoli za upravljanje omrežij SNMP, CLI in CORBA. Za primerjavouporabijo odprtokodni paket Ensuite (Yencap). Kot kazalce za primerjavo protokolov uporabijo podprte funkcionalnosti, število potrebnih transakcij, velikost izmenjanih sporočil in število paketov ter njihovo povprečno velikost, ki jih zahteva posamezen protokol. Z njimi avtorji dobro prikažejo prednosti novega protokola. Zaključujejo, da je NETCONF dobro izhodišče za upravljanje prihajajočih omrežij, ki bodo vse kompleksnejša. Kot izhodišče za nadaljnje delo izpostavljajo predvsem standardizacijo podatkovnega modela omrežij.

V delu [48] so predstavljeni problematika upravljanja omrežij in nova protokola, ki jih je standardiziral IETF, NETCONF [18] in YANG [16]. Avtorji predstavijo svojo rešitev, ki uporablja nove protokole, in jo ovrednotijo. Na

nekaj primerih pokažejo podporo zahtevam, ki so navedene v RFC 3535 [41] ter merijo časovno in prostorsko zahtevnost delovanja. Na koncu poudarijo predvsem, da je glavna vrednost novih standardov v transakcijah in da vsaka transakcija izpolnjuje vse lastnosti ACID (*ang. Atomicity, Consistency, Isolation, Durability*; *Atomarnost, Konsistentnost, Izolacija, Trajnost*) [26]. *Atomarnost* zagotavlja, da se transakcija izvede v celoti, sicer se sploh ne. *Konsistentnost* skrbi, da transakcija napravo prenese iz enega veljavnega stanja v drugo veljavno stanje. *Izolacija* omogoči, da ima vzporedna izvedba dveh sprememb enak učinek, kot če bi bili spremembi izvedeni zaporedno. *Trajnost* pa zagotavlja, da se sprememba po izvedbi tudi shrani.

Z vse večjo zrelostjo novih protokolov se je pojavilo veliko različne programske opreme, ki omogoča napredno upravljanje računalniških omrežij. Avtorji v članku [23] opisujejo zahteve za programsko opremo za upravljanje računalniških omrežij in storitev, ki bo nadomestila trenutno še v večji meri ročno upravljanje omrežij v njihovih podatkovnih centrih. Zapišejo, da je vzdrževanje interno napisanih orodij in skript zelo zapleteno, saj je v primeru nove opreme potrebno ta orodja ustrezno prilagoditi. Izpostavijo, da orodje, ki bi izpolnjevalo vse zahteve, verjetno ne obstaja in da bo za avtomatizacijo njihovega upravljanja potrebnih več rešitev.

Nekateri avtorji so se pristopa lotili drugače in so za lažjo avtomatizacijo omrežij razvili svoje rešitve. Tako Martin Bertolina v članku [11] opiše razvoj sistema za upravljanje računalniških omrežij z uporabo protokola NETCONF [18]. Sistem je razvil kot nadomestilo upravljanja s protkolom SNMP in ukazno vrstico, ki sta postala neprimerna za raven avtomatizacije, ki jo danes potrebujejo omrežja. Za lažje upravljanje je razvil spletni strežnik z uporabniškim vmesnikom in vmesnike HTTP/SOAP za izvajanje osnovnih operacij in nastavitev na napravah. Za implementacijo je uporabil odprtokodne programske knjižnice v jeziku programskem python, kot je ncclient [12], in odprtokodno verzijo orodja OpenYuma [32].

Podobno so se avtomatizacije omrežij lotili tudi strokovnjaki iz grške akademske mreže GRNET, ki so svoje orodje za upravljanje omrežja [34] prav

tako zasnovali okoli protkola NETCONF. Dodatno so za ta namen razvili programsko opremo, imenovano nxpy [37], ki omogoča pretvorbo podatkov XML v objekte programskega jezika Python. Ta omogoča, da za omejen nabor funkcionalnosti protokola NETCONF razvijalcem ni potrebno procesirati dokumentov XML, ampak lahko nastavitve upravljajo z uporabo programskih konstruktov v Pythonu.

2.6 Odprta vprašanja in prihodnji standardi

Standardizacija modelov YANG Z vse bolj razširjeno uporabo modelov YANG se je pokazala tudi potreba po njihovi standardizaciji, ki bo omogočila združljivost med različnimi proizvajalci in sistemi. S tem namenom so v IETF že definirali nekaj modelov, npr. za usmerjanje, vmesnike, liste dostopa [28]. Poleg delovne skupine v IETF se je priprave modelov YANG, ki bi zadostili njihovim potrebam, lotila tudi delovna skupina upravljalcev velikih računalniških omrežij - OpenConfig [3].

Prenova jezika YANG Delovna skupina v IETF (*ang. The Internet Engineering Task Force*) trenutno dela na prenovi jezika za modeliranje YANG [17].

Prihodnji standardi V skupini poteka tudi delo, ki bo omogočilo ločitev jezika za modeliranje YANG od protokola NETCONF. Jezik YANG in modeli YANG so pomembni za predstavitev upravljalske informacije tudi, če se za izmejavo upravljalske informacije med upravljalcem in omrežnim elementom uporabi drug protokol. V ta namen se pripravljajo dodatni protokoli za nastavljanje naprav, ki izhajajo iz NETCONF-a, to sta CoAP [15] in RESTCONF [14]. Za dostavo sprememb se vse bolj uveljavlja tudi protokol gRPC [2].

Poglavje 3

Obstoječe rešitve za upravljanje naprav in storitev

V tem poglavju je predstavljen pregled obstoječih rešitev za upravljanje naprav in storitev. Pregled smo izvedli z namenom ugotoviti, katere obstoječe rešitve bi lahko integrirali v arhitekturo za enovito upravljanje, ki je predstavljena v razdelku 4.

Nekatere rešitve so nastale v odprtokodni skupnosti in industriji kot odgovor na spremenjeno dinamiko in velikost omrežne infrastrukture. Druge so bile prevzete s področja strežniške infrastrukture. Vsaka izmed njih igra pomembno vlogo pri upravljanju omrežne infrastrukture. Glede na njihov namen in funkcionalnosti smo jih v razdelku 3.1 klasificirali v štiri skupine. V razdelku 3.2 je utemeljena izbira rešitev, ki smo jih vključili v predlagano arhitekturo enovite infrastrukturo. Nato so v razdelkih od 3.3 do 3.6 podrobno predstavljene izbrane rešitve.

3.1 Skupine

V razdelku so opisane pomembne skupine programskih rešitev, ki se razvijajo v skupnosti omrežnih operaterjev in v industriji. Vsaka izmed njih predstavlja pomemben sestavni del upravljanja omrežne infrastrukture.

3.1.1 Podatkovni modeli

Trenutno je v svetu upravljanja omrežij še vedno aktualno področje podatkovnih modelov. Njihov pomen smo predstavili v razdelku 2.3, ki opisuje predstavitev upravljalske informacije. Glavni namen delovnih skupin za pripravo podatkovnih modelov je z uporabo jezika YANG definirati sintaktično in semantično predstavitev upravljalske informacije, ki bi jo v isti obliki razumeli vsi omrežni elementi. Na tem področju sta aktivni dve iniciativi, ki združujeta skupnost in industrijo, to sta OpenConfig in delovna skupina v IETF. Obe sta že pripravili nabor modelov YANG [35, 43], ki bodo omogočili enovito predstavitev podatkov za vse omrežne elemente.

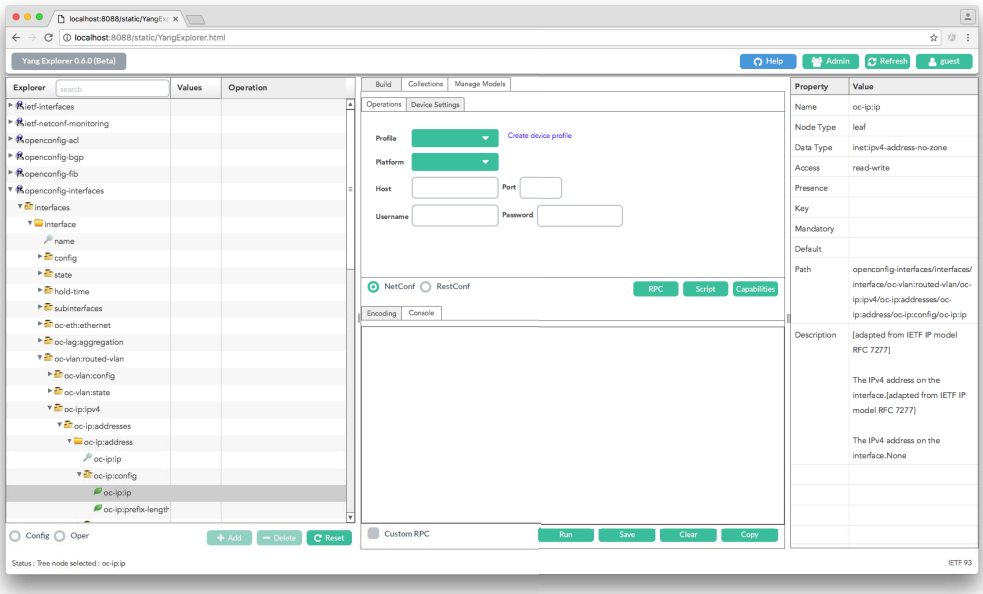
Podpora modelov v končnih napravah je odvisna od proizvajalcev opreme in pritiska skupnosti.

Za hitrejšo vpeljavo modelov v uporabo in lažjo interakcijo z modeli so zato v okviru skupnosti nastala številna orodja. Ločimo jih po naslednjih treh funkcionalnostih [7].

Urejanje modelov omogoča nabor orodij za ustvarjanje in urejanje modelov YANG. Uporabniku nudijo pomoč pri preverjanju sintakse in izgradnji modelov. Primeri takšnih orodij so *yang-vim*, *YANG-IDE*, *YANG design studio*, *YANG Explorer*. Pregled modela YANG z imenom *ietf-interfaces* v orodju *YANG Explorer* je prikazan na sliki 3.1. V levem stolpcu lahko uporabnik pregleduje atribute modela, v desnem pa njihove podrobnosti.

Knjižnice z implementiranimi funkcionalnostmi programskim rešitvam omogočajo lažjo interakcijo z modeli YANG. Razvite so za različne programske jezike, npr. *goyang*, v jeziku Go. *Pyang* in *Pyangbind*, v jeziku Python ter *jYang*, v jeziku Java.

Programska oprema za odjemalce in strežnike omogoča uporabo funkcionalnosti protokolov NETCONF in YANG na odjemalcih in strežnikih. Primera sta *yangcli-pro* in *ConfD Basic*.



Slika 3.1: Prikaz orodja YANG Explorer.

3.1.2 Knjižnice za upravljanje

Knjižnice lahko razdelimo na dva tipa. Prvi tip knjižnic, implementira posamezen protokol za upravljanje omrežnih elementov. Vključujejo podporo za opremo enega ali več proizvajalcev. Komunikacijo s protokolom SNMP omogoča zbirka knjižnic *NET-SNMP*, za protokol NETCONF pa *ncclient*. Za izvajanje klicev REST lahko uporabimo knjižnico v programskem jeziku Python, imenovano *requests*. Delo z ukazno vrstico omrežnih naprav pa poenostavlja knjižnica *netmico*.

Vsi omrežni elementi ne podpirajo naprednih protokolov za programsko upravljanje, kot je NETCONF. Z namenom približati tudi ostale upravljalske protokole zahtevami iz delavnice, ki so predstavljene v razdelku 2.4.1, je nastal drugi tip knjižnic. Primer knjižnice drugega tipa je knjižnica NAPALM [10]. Glavni cilj NAPALM je ponuditi enovit programski vmesnik API do različnih upravljalskih protokolov in jih navzven predstaviti kot funkcionalno enakovredne. S tem ciljem programsko razširja funkcionalnosti protokolov,

kolikor je to mogoče. Razširitev doseže tako, da določene funkcionalnosti programsko emulira.

3.1.3 Orodja za avtomatizacijo upravljanja

Orodja za avtomatizacijo upravljanja so bila prvotno uporabljena za nastavljanje strežniških sistemov in storitev, ki tečejo na njih. Omogočajo upravljanje nastavitev, orkestracijo in avtomatizacijo opravil, ki so jih običajno izvajali sistemski administratorji. Zaradi potrebe po orkestraciji omrežne infrastrukture so začela orodja za avtomatizacijo upravljanja dobivati podporo tudi za upravljanje in nastavljanje omrežnih naprav.

Pri podpori za upravljanje omrežnih elementov so v prednosti tista orodja, ki za potrebe upravljanja na omrežni element ne potrebujejo namestiti nobene programske opreme [6]. Nekatera orodja namreč za svoje delovanje na upravljane naprave in strežnike namestijo posebne odjemalce, s katerimi nato komunicira upravljalec. Orodja, ki za svoje delovanje potrebujejo namestitev dodatnega odjemalca, za upravljanje omrežnih naprav niso primerna, ker nameščanje dodatne programske opreme na omrežne naprave običajno ni mogoče. Operacijski sistemi omrežnih elementov so namreč zelo zaprti in specifični za posameznega proizvajalca.

Orodja so zasnovana zelo razširljivo, kar omogoča številnim operacijskim sistemom in storitve na njih. Podporo lahko razširjajo z uporabo knjižnic *(gl. razdelek 3.1.2)*.

Trenutno sta v skupnosti in industriji zelo uporabljeni ogrodji Puppet [46] in Ansible [27]. Za upravljanje omrežij je zaradi svoje arhitekture najbolj primeren ravno slednji.

3.1.4 Kontrolniki za programsko opredeljena omrežja

Pristop h gradnji omrežij, imenovan programsko opredeljena omrežje (*SDN - Software defined networking*), opisuje omrežja, v katerih sta krmilna in podatkovna ravnina ločeni. Podrobno je pristop k upravljanju programsko

opredeljenih omrežij predstavljen v razdelku 2.2.2.

Krmilna ravnina je centralizirana v obliki krmilnikov SDN, podatkovni pa se izvaja neposredno na fizičnih ali virtualnih omrežnih elementih. Krmilni nivo se odloča o tem, kako bo omrežje delovalo. Upravlja nastavitve in prometne tokove. To doseže z uporabo protokolov, kot so NETCONF, OpenFlow in OVSDB. V industriji in skupnosti se razvija več krmilnikov. Med njimi ima največ finančnih in razvojnih virov krmilnik *OpenDaylight* [4]. Kot so izpostavili avtorji članka [23], je krmilnik zahteven za integracijo. Zelo dobro in poglobljeno poznavanje zahteva tudi za razširanje in razvoj dodatnih funkcionalnosti. Prav tako pomemben predstavnik krmilnikov je Tail-F NCS, ki se trenutno trži pod blagovno znamko Cisco NSO in je plačljiv. Razvijajo se še številni specializirani odprtokodni krmilniki, kot sta *OpenNaaS* in *Floodlight*.

3.2 Izbrane rešitve

V pričujočem razdelku so opisani gradniki, ki smo jih izbrali za zasnovo arhitekture za enovito upravljanje in nastavljanje infrastrukture. Pri izbiri obstoječih rešitev smo se omejili na odprtokodne in brezplačne rešitve. Rešitve smo predhodno v razdelku 3.1 klasificirali v štiri skupine.

Za abstrakcijo upravljalske informacije smo, kjer je to mogoče, izbrali modele OpenConfig, ki imajo za seboj izkušnje velikih upravljalcev omrežnih infrastruktur in bodo v prihodnjosti podpirali tudi paradigmo pretočne telemetrije, ki bo poenostavila nadzor omrežij. Pomembna je tudi zaveza velikih proizvajalcev omrežne opreme, kot sta Juniper in Cisco, da bodo v svojih omrežnih napravah dodali podporo za modele OpenConfig. OpenConfig smo predstavili v razdelku 3.3.

Ker bomo našo rešitev zasnovali in razvili v programskem jeziku Pyton, smo za interakcijo z modeli YANG in njihovo preverjanje uporabili knjižnici pyang [30] in pyangbind [44]. Podrobneje sta predstavljeni v razdelku 3.4.

Za upravljanje smo v svojo rešitev vključili orodje za avtomatizacijo na-

stavljanje in upravljanje, Ansible. Le to ima najboljšo podporo [6] za upravljanje omrežnih naprav, saj za upravljanje ne zahteva nameščanja dodatne programske opreme na omrežno napravo. Ansible je zelo razširljiv in ima veliko uporabnikov. Dolgoročno vzdržnost rešitve izboljšuje zaveza proizvajalca orodja Ansible, RedHat-a, ki se je zavezal k pospešenemu razvoju funkcionalnosti za upravljanje omrežnih naprav. Poleg tega nam je uporaba Ansible močno poenostavila upravljanje omrežnih storitev, ki tečejo na Linux strežnikih, saj je bila to njegova primarna naloga. Podrobno je Ansible opisan v razdelku 3.6.

Da smo lahko zagotovili upravljanje heterogenih omrežnih naprav, med katerimi mnoge ne podpirajo najsodobnejših metod in protokolov za upravljanje, smo za nastavljanje omrežnih elementov uporabili knjižnico NAPALM. Ta omogoča integracijo z Ansiblom in podpira veliko različne omrežne opreme. NAPALM je podrobno predstavljen v razdelku 3.5.

3.3 OpenConfig

OpenConfig [3] je neformalna skupina upravljalcev velikih računalniških omrežij, kot so so Google, Facebook, Microsoft in Yahoo. Skupaj poskušajo omrežno infrastrukturo spraviti na nivo strežniške in jo narediti bolj dinamično, programabilno in prilagojeno na programsko določena omrežja. Skrbijo, da bodo vse operacije in nastavitve v omrežjih definirane z uporabo modelov. Njihov glavni cilj je priprava modelov YANG na podlagi zahtev večih operaterjev.

Njihovi modeli YANG [43] gradijo na modelih, ki jih pripravlja IETF in jih razširjajo. Poleg nastavljanja skrbijo tudi za nadzor omrežij, ki se imenuje pretočna telemetrija (*ang. streaming telemetry*) Njen namen je izboljšava trenutno pogosto uporabljenega protokola SNMP, ki zaradi slabe povečljivosti in razširljivosti potrebuje izboljšave.

3.4 Pyang in Pyangbind

Pyang [30] je knjižnica za delo z modeli YANG za programski jezik Python. Vključuje tudi orodje za delo v ukazni vrstici.

Podpira preverjanje modelov YANG, pretvorbo modelov YANG v XML predstavitev modelov, imenovano YIN. Omogoča, da modele YANG prevedemo v DSDL sheme, ki jih lahko uporabljamo za validacije različnih instanc XML dokumentov. Omogoča tudi generiranje drevesnega prikaza YANG modelov in UML diagramov in nudi pretvorbo instanc modelov YANG iz XML v JSON.

Podpira specifikacije jezika YANG, ki so zapisane v dokumentih: *RFC 6020*, *RFC 6087*, *RFC 6110*, *RFC 6643*.

PyangBind je vtič za Pyang [44], ki je nastal v okviru delovne skupine OpenConfig. Omogoča generiranje razredov jezika Python na podlagi hiearhije modela YANG. Razrede, ki jih ustvari, lahko neposredno uporabljamo za interakcijo v programskem jeziku Python. Omogoča tvorjenje novih instanc modelov YANG z manipulacijo neposredno v programskem jeziku Python. Te instance lahko nato porazporedimo v formate, ki jih lahko shranimo ali izmenjamo z omrežnim elementom (npr. JSON ali XML). Omogoča tudi, da v objekt uvozimo nove instance modelov YANG, ki jih lahko nato spreminjamo.

3.5 NAPALM

NAPALM [10] je knjižnica v programskem jeziku Python, ki služi kot abstrakcijski nivo za avtomatizacijo in programabilnost omrežij s podporo za različne proizvajalce omrežne opreme. Za upravljanje naprave posameznega proizvajalca uporablja enega izmed protokolov, ki smo jih predstavili v poglavju 2. Programska oprema, ki vsebuje podporo za tip omrežne opreme, se imenuje gonilnik, metoda, ki jo uporablja za upravljanje naprave, pa je tista, ki je najbolje podprta na napravi. Za posamezne naprave nudi enovit API za upravljanje, ne glede na to, katero metodo uporablja gonilnik. Omogoča spre-

minjanje nastavitev in pridobivanje podatkov o trenutnem stanju omrežnih elementov.

V splošnem podpira spodaj navedene funkcionalnosti. V primeru, da operacijski sistem na napravi katere od teh funkcionalnosti ne podpira, jih gonilnik emulira *(gl. razdelek 2.4.8)*.

- **Config replace:** Zamenjava nastavitve na napravi z novo.
- **Config merge:** Združitev novih nastavitev z obstoječimi.
- **Compare config:** Primerjava obstoječih in novih nastavitev, ki jih želimo naložiti na napravo.
- **Atomic Changes:** Podpora za atomarnost sprememb.
- **Rollback :** Možnost, da v primeru napake omrežni element vrnemo v stanje, v kakršnem je bilo prej.
- **Get Facts:** Pridobivanje podatkov o operativnem stanju in nastavitvah naprave.

NAPALM vsebuje gonilnike za operacijske sisteme na omrežni opremi: eos, junos, iosxr, fortios, ibm, nxos, ios, pluribus, panos in omogoča integracijo z ogrodji za avtomatizacijo nastavljanja, kot sta Ansible in Salt.

3.6 Ansible

Ansible [27] je odprtokodno orodje za avtomatizacijo nastavljanja in upravljanja naprav in storitev. Omogoča tudi orkestracijo nalog, ki se morajo izvesti na več strežnikih in napravah hkrati. Za upravljanje omrežnih elementov in strežnikov nanje ni potrebno nameščati dodatne programske opreme [6]. V tem se razlikuje od drugih popularnih orodij za upravljanje, kot so Chef, Puppet in CFEngine. Njegova uporaba je tako enostavnejša in primernejša za uporabo pri upravljanju omrežnih elementov, na katere običajno ni mogoče namestiti dodatne programske opreme.

Spodaj so predstavljeni pomembni sestavni deli orodja Ansible.

- **Inventorij (*ang. inventory*):** Vsebuje vse naprave, ki jih upravlja. Lahko je v obliki tekstovne datoteke ali tudi programski vmesnik, ki vrne inventorij v obliki JSON.
- **Vloge (*ang. roles*):** Vsaka vloga je sestavljena iz nalog. Znotraj naloge so podrobno definirani vsi potrebni parametri za izvajanje naloge. Za vsakega gostitelja definiramo, katere vloge izvaja. Preprost primer vloge je npr. nudenje storitve DHCP.
- **Datoteke Playbook (*ang. playbook*):** Vsebujejo predstavitev nastavitev in upravljanja v Ansiblu. Predstavljene so v formatu YAML. Datoteke Playbook skupinam posameznih gostiteljev dodeljujejo vloge.
- **Moduli (*ang. module*):** Moduli v Ansiblu opravljajo delo. Lahko so implementirani v različnih skriptnih jezikih, npr. Pythonu. Pomembno je, da izpolnjujejo zahtevo po idempotentnosti, kar pomeni, da v primeru ponovitev operacije upravljani sistem vedno spravi v isto željeno stanje.
- **Vtičniki (*ang. plugin*):** Razširjajo osnovne funkcionalnosti Ansibla. Takšni primeri so dodatek beleženja, dodatne metode za komunikacijo s strežniki in omrežnimi elementi ipd.

Poglavje 4

Razvoj infrastrukture

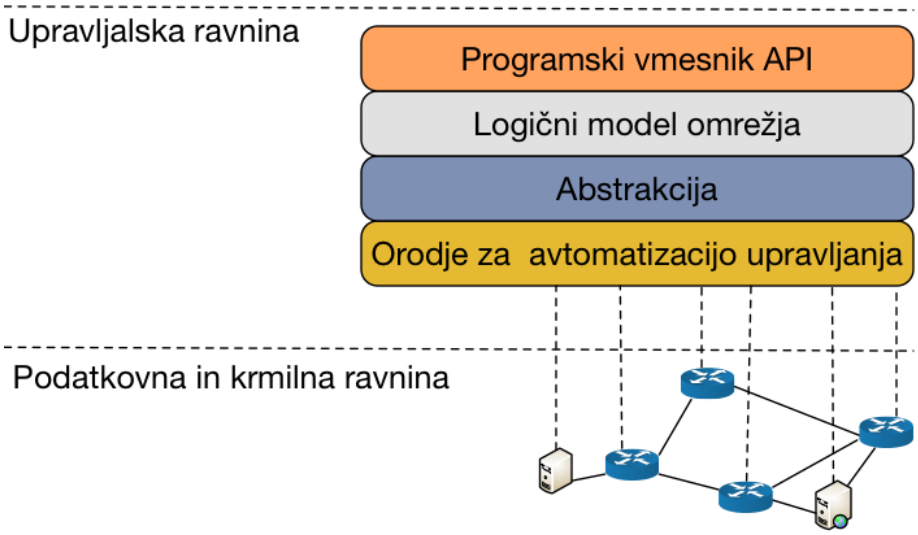
V tem poglavju je predstavljena arhitektura infrastrukture za enovito nastavljanje in upravljanje računalniških omrežij in storitev. Rešitev smo zasnovali na podlagi zahtev upravljalcev velikih omrežij, ki smo jih navedli v razdelku 1.1, in raziskave metod ter protokolov, ki smo jih ovrednotili v poglavju 2. Pri razvoju smo uporabili obstoječe rešitve, predstavljene v poglavju 3. Stremeli smo k modularnosti in razširljivosti naše rešitve, saj je zelo pomembno, da omogoča dodajanje novih funkcionalnosti in podpore za različne storitve.

V razdelku 4.1 so predstavljeni zasnova rešitve, uporabljene tehnologije ter sestavni deli rešitve. V razdelku 4.2 je opisan predlog za nadaljni razvoj novih funkcionalnosti in storitev. Nadalje je v razdelku 4.3 prikazan primer uporabe in koraki, skozi katere se želene spremembe uporabnika odrazijo na omrežni infrastrukturi. Predlagali smo tudi, kako rešitev umestiti v produkcijsko omrežje in kako zagovoti najvišji nivo varnosti.

Izvorna koda rešitve in dokumentacija sta na voljo na repozitoriju git na naslovu https://github.com/blazdivjak/automator.

4.1 Arhitektura

V razdelku 4.1.1 je predstavljena zasnova arhitekture in v razdelku 4.1.2 njeni sloji.



Slika 4.1: Arhitektura enovite infrastrukture za upravljanje računalniških omrežij in storitev.

4.1.1 Zasnova in uporabljene tehnologije

Predlagana arhitektura enovite infrastrukture za upravljanje in nastavljanje omrežnih naprav in storitev *(gl. slika 4.1)* je razdeljena na več slojev. Zasnovali smo jo okoli orodja za avtomatizacijo upravljanja in nastavljanja, Ansible *(gl. razdelek 3.1.3)*.

Orodje Ansible smo razširili z uporabo knjižnice NAPALM *(gl. razdelek 3.5)*, ki nudi enovit programski vmesnik API za upravljanje širokega nabora omrežnih elementov.

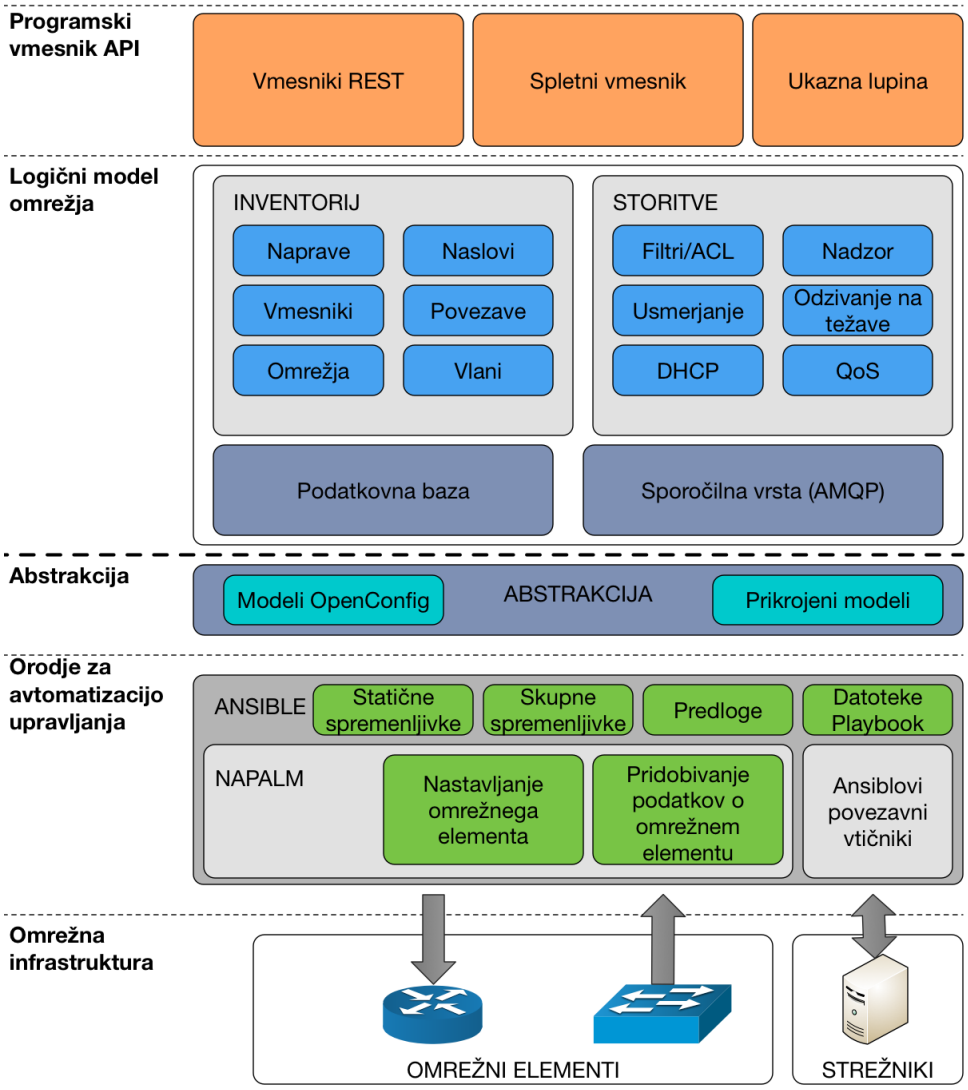
Orodju Ansible smo dodali dodaten sloj abstrakcije, s pomočjo katerega lahko nastavitve omrežnih elementov predstavimo z uporabo modelov YANG, ki jih pripravlja delovna skupina OpenConfig *(gl. razdelek 3.3)*.

Za poenostavitev interakcije z omrežno infrastrukturo smo pripravili logično predstavitev omrežne infrastrukture. Za implementacijo smo uporabili programski jezik Python [5] in ogrodje za razvoj spletnih aplikacij Django [1]. Logična predstavitev omrežne infrastrukture omogoča zapis nastavitev in stanja omrežja v podatkovno bazo. Obenem je upravljanje omrežja z uporabo logičnega modela lažje kot neposredno spreminjanje podatkov z uporabo podatkovnih modelov YANG. Razumevanje modelov YANG namreč zahteva dobro poznavanje omrežnih naprav in storitev.

Za interakcijo z logičnim modelom omrežja smo pripravili vmesnike REST, ki omogočajo upravljanje omrežne infrastrukture iz aplikacij in skript. Za implementacijo smo uporabili knjižnico Django REST Framework [22]. Upravljanje je mogoče tudi s spletnega vmesnika (*ang. Django Admin*) in ukazne lupine (*ang. Django Shell*).

4.1.2 Sloji arhitekture

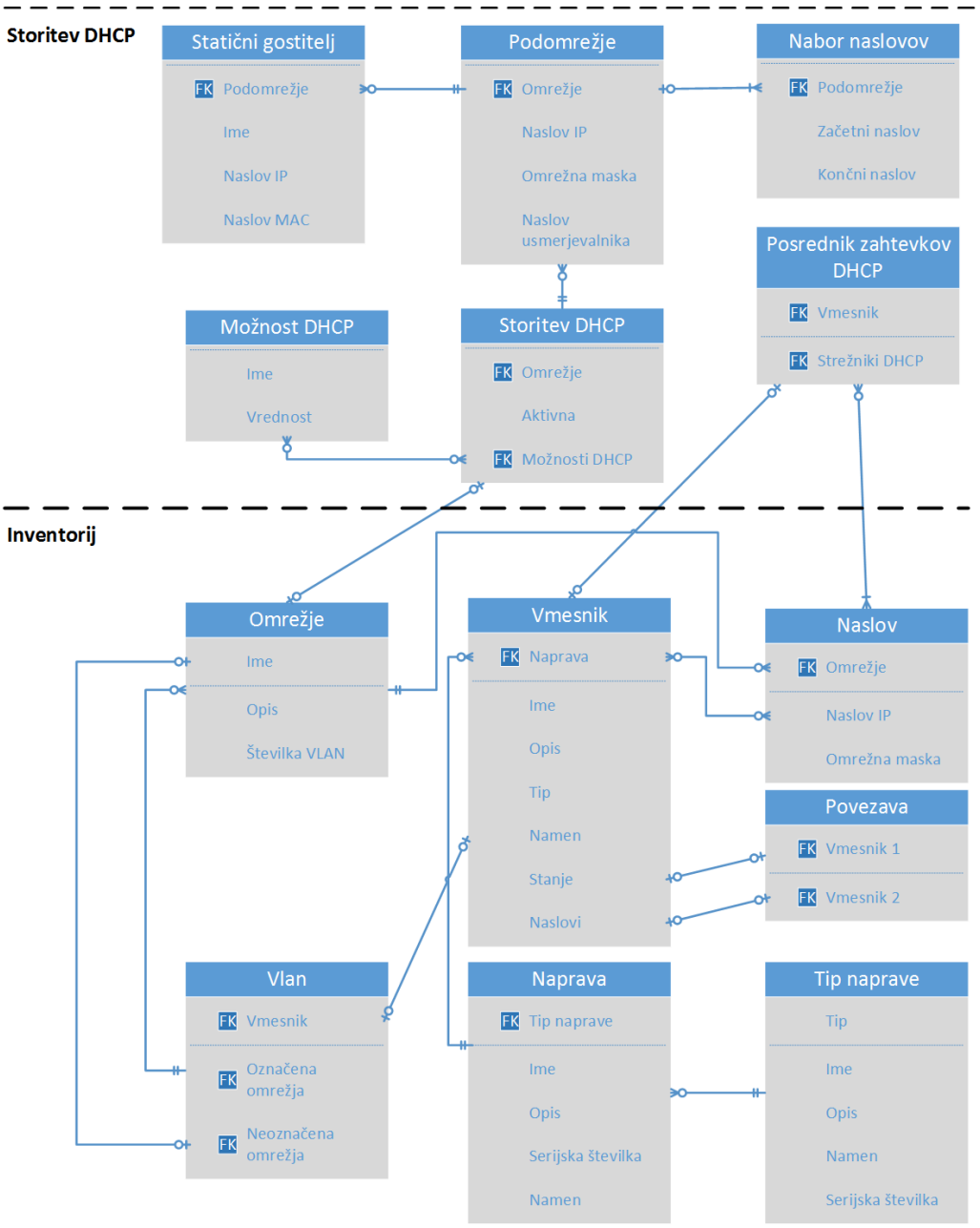
V naslednjih odstavkih so predstavljeni sloji, ki sestavljajo rešitev in njihov pomen za njeno delovanje. Kako se združijo v celoto, da izpolnjujejo svojo funkcionalnost, je predstavljeno na primeru v razdelku 4.3.



Slika 4.2: Sestavni deli enovite infrastrukture za upravljanje računalniških omrežij in storitev.

Programski vmesnik API omogoča upravljanje izbranih funkcionalnosti omrežne infrastrukture. Programski vmesnik API je sestavljen iz vmesnikov REST, ki so namenjeni upravljanju infrastrukture iz aplikacij ali skript. Ukazna lupina nudi možnost za avtomatizacijo pogostih opravil neposredno na sistemu, na katerem je celotna rešitev nameščena. Za pregled nad trenutnimi podatki v bazi je omrežnemu tehniku na voljo spletni vmesnik, ki je del sistema Django. Preko spletnega vmesnika lahko tehnik tudi upravlja z infrastrukturo.

Logični model omrežja je del rešitve, s katerim rokuje uporabnik. Uporabnik logičnemu modelu z uporabo programskega vmesnika API sporoči želeno stanje infrastrukture. Logični model je sestavljen iz inventorija in nabora omrežnih storitev *(gl. slika 4.3)*. Inventorij združuje osnovne omrežne funkcije in vsebuje spisek vseh naprav ter njihovih tipov. Omogoča shranjevanje vmesnikov na napravah in povezav med njimi. V tabeli *Vlan* omogoča tudi shranjevanje nastavitev vlanov na fizičnem vmesniku. Model vsebuje podatke o omrežjih in naslovnem prostoru. Drugi sestavni del modela so omrežne storitve. V logičnem modelu so implementirane storitve, katerih nastavitve se pogosto spreminjajo in njihovo upravljanje želimo omogočiti tudi uporabnikom. Kot primera storitev, ki smo ju preizkusili v poglavju 6, smo implementirali storitvi dinamičnega dodeljevanja naslovov IP (*DHCP*) in storitev za odzivanje na težave v omrežju. Storitve DHCP ima tudi logični model, saj se nastavitve storitve veliko spreminjajo, storitev za odzivanje na težave pa logičnega modela nima, saj se privzeto vključi brez dodatnega nastavljanja. Model storitve DHCP uporabniku omogoča, da za posamezno podomrežje v omrežju izbere nabore naslovov IP, ki bodo dodeljeni uporabnikom. V tabelo *Statični gostitelj* lahko shranimo podatek o tem, katere naprave bodo v omrežju vedno dobile isti naslov IP. Posrednik zahtevkov DHCP pa omogoča, da za posamezen vmesnik na omrežni napravi nastavimo, na kateri strežnik DHCP naj se posredujejo zahtevki DHCP v omrežju. Vsi podatki logičnega modela so zapisani v podatkovni bazi (npr. MySQL). Vse



Slika 4.3: Logični model.

spremembe, ki jih izvedejo uporabniki, se shranijo v podatkovni bazi in se nato asinhrono izvedejo na infrastrukturi. Vsaka sprememba se zato v obliki sporočila zapiše v sporočilno vrsto (npr. RabbitMQ) in se nato periodično obdela glede na nastavitev zakasnitve.

Abstrakcija nastavitev omogoča, da nastavitve, ki jih zgenerira sistem, razumejo omrežni elementi in storitve različnih proizvajalcev. V rešitvi smo za potrebe demostracije uporabili modele OpenConfig za vmesnike (*ang. interfaces*), posredovanje zahtevkov DHCP (*ang. relay-agent*) in vlan. Omrežna oprema, ki je bila na voljo v laboratoriju, modelov OpenConfig še ne podpira, zato jih na njižjem sloju z uporabo predlog prevedemo v obliko, ki jo razume posamezna naprava.

Orodje za avtomatizacijo upravljanja omogoča izvedbo nastavljanja in upravljanja naprav in storitev. Za njegovo implementacijo smo uporabili ogrodje Ansible. Ta sloj omogoča možnosti definiranja vlog za nastavljanje omrežnih elementov in storitev. Nudi možnost definicije statičnih spremenljivk, ki jih lahko uporabimo za nastavitev omrežnih elementov ali storitev. Omogoča prevajanje nastavitev iz reprezentacije v modelih OpenConfig v predstavitev, ki jo razume posamezna naprava ali storitev. Za interakcijo z omrežnimi elementi uporablja vtičnik NAPALM, za upravljanje storitev, ki tečejo na strežnikih Linux, pa privzeti povezavni vtičnik paramiko.

Omrežna infrastruktura je sestavljena iz različnih naprav (npr. usmerjevalniki, stikala, strežniki itd.), ki za upravljanje s sistemom ne potrebujejo biti dodatno prilagojene. Vsaka naprava celostno opravlja svojo funkcijo in za delovanje ni neposredno odvisna od sistema za upravljanje. Sistem napravo le ustrezno nastavi in na njej izvaja upravljalske naloge *(gl. razdelek 2.1)*, v realnem času pa nikakor ne vpliva na prepošiljanje paketkov.

4.2 Dodatek nove omrežne storitve

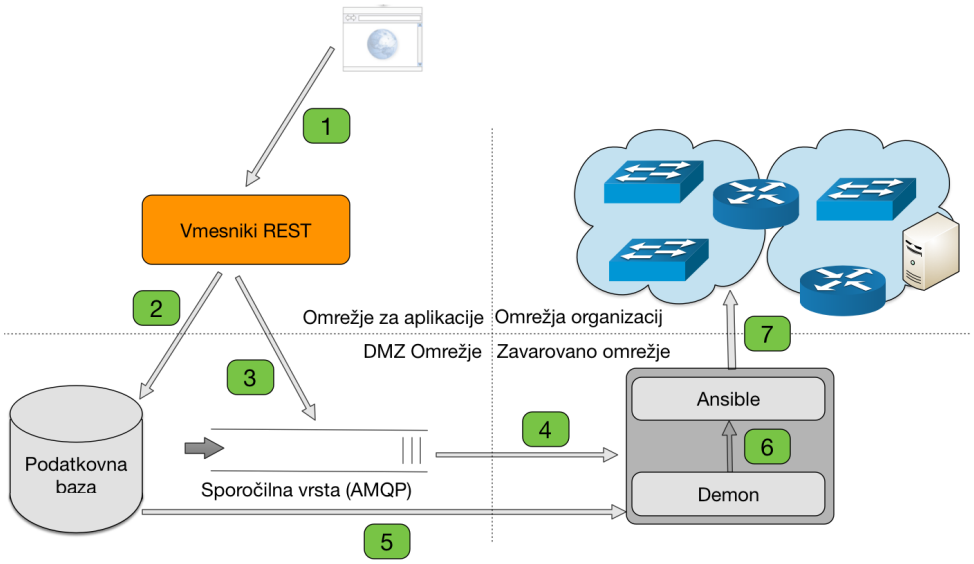
V tem razdelku predlagamo postopek za razširjanje arhitekture in vključevanje novih storitev v sistem. Zaradi modularnosti arhitekture lahko upravljanje nove omrežne storitve v sistem vpeljemo postopoma. To zmanjša zahtevnost razvoja in vpeljave ter omogoča razvoj z uporabo agilnih metod razvoja programske opreme. Arhitektura omogoča, da storitve implementiramo kot vloge (*ang. roles*) v orodju Ansible *(gl. razdelek 3.6)*. Za predstavitev izberemo ustrezen podatkovni model, v kolikor gre za omrežne naprave, predlagamo uporabo obstoječih modelov OpenConfig. V primeru, da se nastavitve storitev veliko spreminjajo in bi radi spreminjanje nastavitev omogočili uporabnikom, pripravimo ustrezni logični model storitve, s katerim bo mogoče nastavitve zapisati v podatkovni bazi. Nato definiramo vmesnike REST, preko katerih bo mogoče spreminjanje nastavitev iz skript in uporabniških vmesnikov.

4.3 Primer uporabe

V razdelku je predstavljeno, kako sistem za enovito upravljanje vključiti v obstoječo infrastrukturo. Na primeru nastavitve omrežnega elementa je prikazano, kako se sloji sistema povezujejo med seboj med izvajanjem.

Za zagotavljanje večje varnosti in modularnosti se lahko različne komponente sistema za enovito upravljanje namesti na različne strežnike. Strežniki so s takim pristopom umeščeni v omrežja z različnim nivojem varnosti. Sistem za upravljanje sestavlja pet komponent. Zaledni strežnik z vmesniki REST, podatkovna baza, sporočilna vrsta, demon za procesiranje sprememb in orodje Ansible. Na sliki 4.4 je prikazana priporočena umestitev komponent infrastrukture v omrežja.

Razdelitev je pomembna predvsem z varnostnega vidika. Zaledni strežnik z vmesniki REST, do katerega dostopajo spletne aplikacije ali skripte, umestimo v omrežje za aplikacije, saj so njegove storitve dostopne od zunaj. Priporočljivo je, da je to omrežje varovano z dostopnimi listami ali požarno



Slika 4.4: Primer zahtevka za spremembo nastavitev omrežne naprave ali storitve.

pregrado in dopušča dostop le do vrat, na katerih so vmesniki REST. Podatkovna baza in sporočilna vrsta sta umeščeni v demitalizirano omrežje (DMZ). Iz tega omrežja nimata dostopa v druga, bolj varovana omrežja in le nudita svoje storitve drugim zalednim sistemom. Priporočljivo je varovanje z uporabo požarne pregrade in dostopnih list. Dostop do vrat, na katerih tečeta podatkovna baza in sporočilna vrsta, mora biti dovoljen le za zaledni sistem z vmesniki REST in demona, ki procesira spremembe. V najbolj varovano omrežje sta umeščena demon, ki procesira sporočila o spremembah infrastrukture, in orodje Ansible, ki ima dostop do vse omrežne infrastrukture, ki jo upravlja. Omrežni elementi in storitve so umeščeni v omrežja z različnim nivojem varovanja in v splošnem nimajo dostopa do nobenega izmed prej naštetih komponent infrastrukture za upravljanje in nastavljanje.

Za lažjo predstavo, kako se sloji infrastrukture med seboj povezujejo, slika 4.4 prikazuje postopek, ki se izvede, ko uporabnik v aplikaciji zahteva spremembo na infrastrukturi. S pomočjo slike bomo sledili korakom, ki se izvedejo, ko vmesniku stikala ge-0/0/1 nastavimo 3 omrežja s številkami vlan 2,4 in 5.

1. Uporabnik z uporabo aplikacije preko vmesnikov REST v sistem sporoči želeno spremembo. Sistemu ukaže nastavitev nastavitev vlanov s številkami 2,4 in 5 za vmesnik ge-0/0/1 na stikalu.

2. Spremembe se na logičnem nivoju zapišejo v podatkovno bazo in uporabnik z nastavljanjem na tem mestu zaključi, za nadaljne korake poskrbi sistem.

3. V sporočilno vrsto se dodajo sporočila, ki sporočajo, da je potrebna posodobitev nastavitev na napravi.

4. Sistem za upravljanje, ki teče na zalednem strežniku, je sestavljen iz demona za procesiranje sporočil in orodja Ansible. Demon periodično procesira sporočila iz sporočilne vrste. Ko prejme sporočilo za posodobitev nastavitev na omrežni napravi, sproži nalogo, ki to izvede.

5. Naloga iz podatkovne baze pridobi najnovejše podatke in jih z uporabo abstrakcijskega sloja pretvori v obliko definirano v modelu YANG ali lastniškem modelu, ki smo ga definirali za storitev ali funkcionalnost, če model YANG ni na voljo.

   Izvoz nastavitev vmesnika, ki so shranjene v inventoriju v obliko modela OpenConfig. Nastavitev vmesnika je prikazana v obliki YAML in definira vse potrebne podatke za nastavitev.

```
interfaces:
  interface:
    ge-0/0/1:
      config:
        description: --eduroam access point--
        type: ethernetCsmacd
      ethernet:
        switched-vlan:
          config:
            interface-mode: TRUNK
            native-vlan: 4
            trunk-vlans:
            - 2
            - 5
```

6. Naloga izvede datoteke playbook v orodju Ansible. V datotekah playbook so definirana pravila za generiranje nastavitev in njihovo namestitev na omrežne elemente in strežnike. Trenutno se pri generiranju nastavitev uporabljajo predloge Jinja2, ki reprezentacijo nastavitev v instancah modelov OpenConfig pretvorijo v nastavitve, ki jih razumejo omrežni elementi. Podpora za modele OpenConfig namreč v omrežne elemente šele prihaja.

   Izsek Jinja2 predloge, ki se jo uporablja za pretvorbo instance modela OpenConfig v obliko za operacijski sistem JunOS.

```
interfaces {
{% for _, interface in interfaces.interface|dictsort() %}
    {{ interface.name }} {
{% if '.' in interface.name and 'routed-vlan' in interface %}
        unit {{ interface['routed-vlan'].config.vlan }} {
{% else %}
        unit 0 {
```

```
{% endif %}
{% if interface.config.description %}
          description "{{ interface.config.description }}";
{% endif %}
{% if interface.config.enabled|default(True)==False %}
          disable;
{% endif %}
{% if 'ethernet' in interface and 'switched-vlan' in interface.ethernet %}
{% set vlan = interface.ethernet['switched-vlan'] %}
          family ethernet-switching {
              port-mode {{ vlan.config['interface-mode']|lower }};
{% if 'access-vlan' in vlan.config %}
              vlan{
                  members {{ vlan.config['access-vlan'] }};
              }
{% endif %}
```

Primer konfiguracijske datoteke, ki jo razume stikalo proizvajalca JunOS.

```
interfaces {
...
    ge-0/0/1 {
        unit 0 {
            description "--eduroam access point--";
            family ethernet-switching {
                port-mode trunk;
                native-vlan-id 4;
                vlan{
                    members [ 2,5 ];
                }
            }
...
}
```

7. Orodje Ansible z uporabo povezavnih vtičnikov in pripravljenimi nastavitvami izvede spremembe ali pridobi podatke o operativnem stanju omrežnih elementov in strežnikov, na katerem tečejo storiteve. Glede na uspeh sprememb se shrani rezultat obdelave naloge. V primeru neuspeha se lahko naloga ponovi ali o rezultatu obvesti človeškega operaterja.

Poglavje 5

Ovrednotenje predlagane arhitekture

V tem poglavju smo ovrednotili predlagano arhitekturo. Za potrebe ovrednotenja smo vzpostavili laboratorijsko omrežje, s katerim smo se želeli kar se da približati realnemu omrežju. Laboratorijsko omrežje smo podrobno predstavili v razdelku 5.1.

Delovanje infrastrukture smo preizkusili tako, da smo na podlagi potreb upravljalcev pripravili dva scenarija, s katerima se upravljalci omrežne infrastrukture pogosto srečujejo. Izzivi upravljalcev in njihove potrebe so opisane v sledečih odstavkih, pripravljena scenarija pa v razdelkih 5.2.1 in 5.2.2.

V razdelku 5.3 smo ovrednotili preizkušano infrastrukturo in v razdelku 5.4 podali izhodišča za nadaljne delo.

Razširitev omrežja Med vzpostavitvijo ali prenovo omrežij se v omrežju pojavijo novi omrežni elementi, ki za svoje delovanje še niso ustrezno nastavljeni. Da omrežje in njegove storitve začnejo funkcionirati, morajo človeški operaterji vsak omrežni element posebej nastaviti ročno ali s pomočjo skripte. Nastavljanje običajno poteka z uporabo ukazne vrstice. Nastavitve na omrežnih elementih se po določenem časovnem obdobju začnejo močno razlikovati med seboj. Temu botruje več razlogov: zahteve uporabnikov, različni opera-

terji in človeške napake pri vnosu nastavitev. Nastaviti morajo tudi podporne storitve, ki lahko tečejo na omrežnih elementih ali centraliziranih strežnikih. Takšno delo je zamudno in ob večjih priklopih zahteva veliko človeških virov ter je pogosto vzrok nekonsistentnih nastavitev.

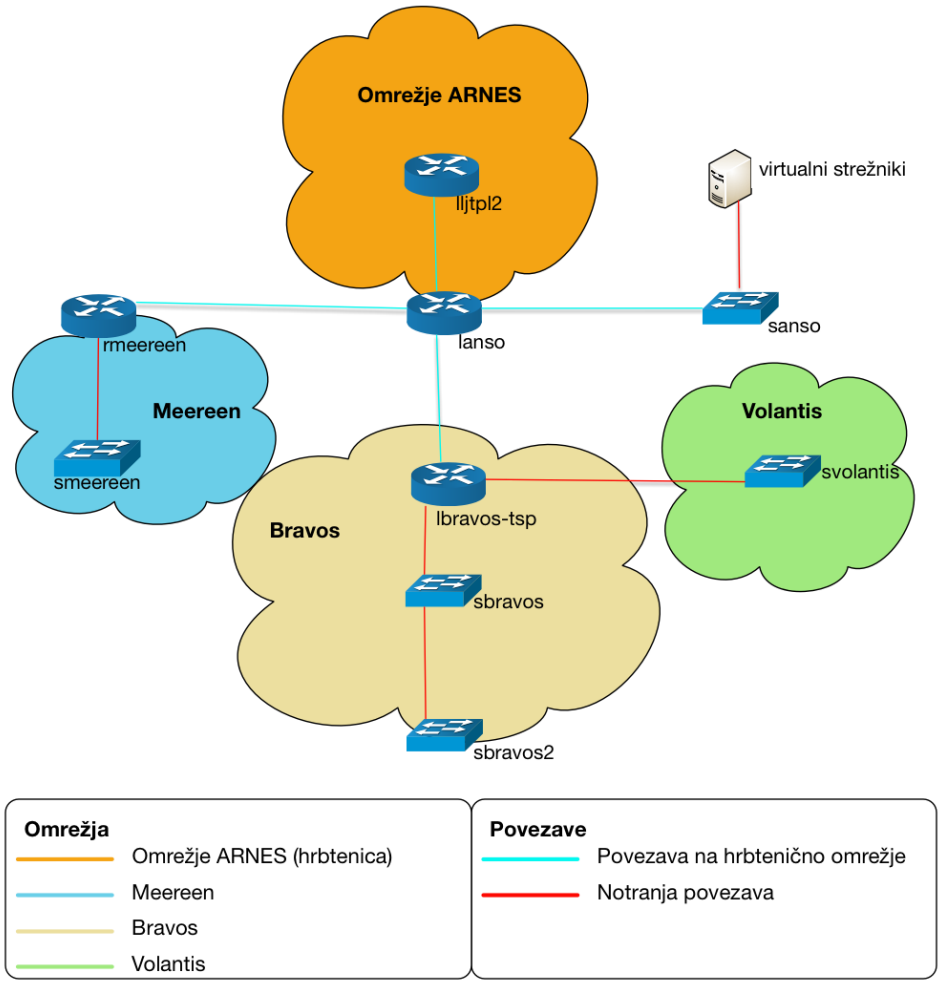
Trenutno stanje bi radi izboljšali z uporabo enovite infrastrukture, ki bo omogočala hitrejše dodajanje novih omrežnih elementov. Pomagala bo standardizirati nastavitve in zagotoviti, da se le te izvedejo zanesljivo. Obenem bi morala enovita infrastruktura nastaviti tudi vse potrebne podporne storitve, ki jih omrežje potrebuje za svoje delovanje. Nabor storitev se lahko skozi čas razširja. Poleg tega bi radi zmanjšali količino virov, potrebnih za upravljanje omrežja, zato bi urejanje nekaterih nastavitev prepustili uporabnikom.

Odzivanje na probleme v omrežju Med delovanjem omrežne infrastrukture pride do različnih dogodkov. Trenutno jih upravljalci zaznavajo na različne načine. Pogosto uporabljajo protokol SNMP za pridobivanje operativnega stanja omrežnih elementov. Na podlagi podatkov jih nato o nepravilnostih obvestijo nadzorni sistemi. Dodatno lahko stanje omrežnih storitev spremljajo z pregledovanjem dnevniških datotek, ki jih izpisujejo storitve. V obeh primerih mora odziv na obvestilo izvesti človeški operater, ki bodisi poseže v nastavitve omrežnih elementov bodisi v storitev.

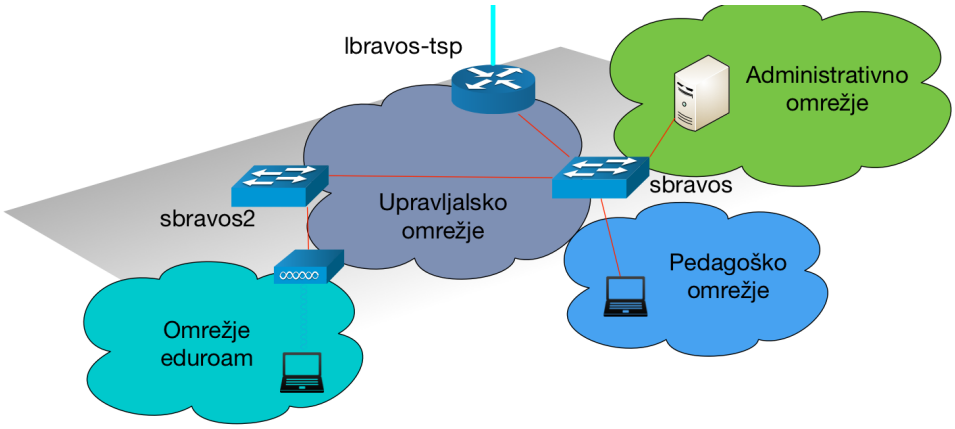
Stanje bi upravljalci omrežij radi izboljšali in se na določene probleme v omrežju odzvali avtomatsko, brez posega človeškega operaterja. Ob tem bi se bodisi ustrezno prilagodilo nastavitve omrežnih elementov bodisi storitev. S tem bi radi doseglji krajši čas izpada storitve in dinamično prilagajanje storitev glede na potrebo.

5.1 Laboratorij

Za potrebe preizkušanja tehnologij in ovrednotenja predlagane arhitekture smo vzpostavili laboratorijsko omrežje na Akademski in raziskovalni mreži Slovenije v nadaljevanju ARNES *(gl. slika 5.1*). Uporabili ga bomo za prikaz



Slika 5.1: Preizkusni laboratorij na Arnesu.



Slika 5.2: Notranje omrežje organizacije Bravos.

preizkusnih scenarijev. Z njim smo se poskusili povsem približati realnemu omrežju, ki deluje v slovenskem akademskem in raziskovalnem prostoru in med seboj povezuje približno 1500 organizacij.

Preizkusno omrežje je sestavljeno iz dela hrbteničnega omrežja, poimenovanega omrežje ARNES, in treh organizacij, ki so vključene v omrežje ARNES. Omrežna infrastruktura posamezne organizacije *(gl. slika 5.2)* je sestavljena iz več omrežnih elementov, stikal in usmerjevalnikov, ki so povezani na hrbtenično omrežje. Vsaka organizacija ima več različnih omrežij, ki so razdeljeni po namembnosti. To so pedagoško, administrativno, upravljalsko in eduroam. Naslovni prostor v omrežjih vsebuje oba tipa protokolov IP. IPv4 in IPv6.

Za potrebe laboratorija smo v hrbtenično omrežje vključili še virtualizacijsko infrastrukturo, na kateri smo lahko poganjali strežnike za omrežne storitve.

5.2 Preizkusni scenariji

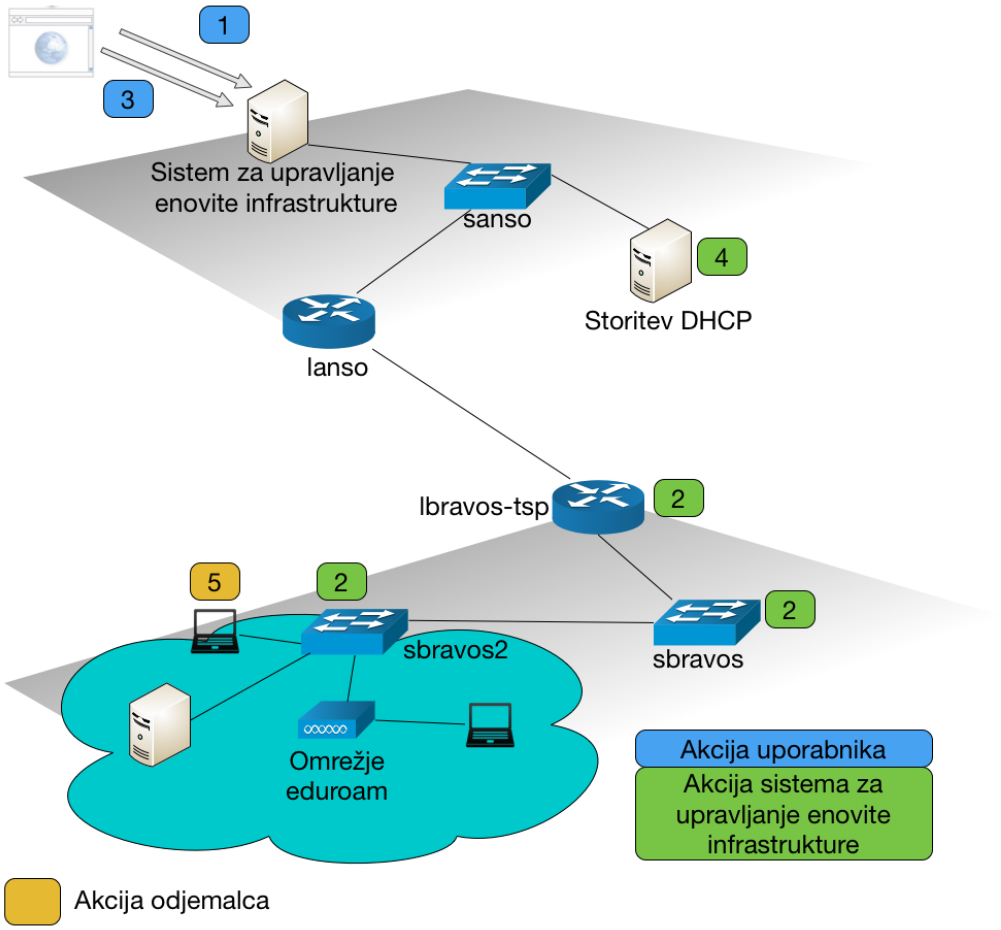
Implementacijo predlagane infrastrukture smo preizkusili na dveh scenarijih, ki smo jih pripravili na podlagi potreb upravljalcev velikih omrežij in problemov, zapisanih v uvodnem poglavju. Prvi scenarij, opisan v razdelku 5.2.1, obsega nastavitev omrežnih elementov in storitve v omrežju. Drugi scenarij, opisan v razdelku 5.2.2, pa predstavlja, kako predlagana infrastruktura z uporabo podatkov o delovanju omrežja in storitev omogoča prilagajanje omrežnih nastavitev za zagotavljanje najvišjega nivoja delovanja.

5.2.1 Razširitev omrežja

V prvem scenariju smo razširili omrežje organizacije Bravos *(gl. slika 5.3)*, ki smo mu dodali še omrežje za brezžične uporabnike z imenom eduroam. Za ta namen smo v omrežje vključili dodatno stikalo sbravos2, na katerega bodo priključene dostopovne točke za brezžično omrežje in naprave, ki so v omrežje priključene s kablom. Poleg tega smo za poenostavitev dodeljevanja naslovnega prostora vključili in nastavili tudi storitev samodejnega dodeljevanja naslovov IP (*Dynamic host configuration protocol*) v omrežju eduroam organizacije Bravos.

Razširitev izvedemo tako, da v logični model inventorija in storitve DHCP vnesemo vse potrebne spremembe. Nato sistem za upravljanje na podlagi vnešenih podatkov avtomatsko nastavi omrežno in storitveno infrastrukturo. Izvedba je podrobno opisana v sledečih korakih *(gl. slika 5.3)*.

1. Za vmestitev novega stikala v omrežje in dodatek omrežja eduroam v logični model inventorija shranimo podatke, ki so prikazani v tabeli 5.1. Za vnos podatkov uporabimo inventorijev programski vmesnik API *(gl. slika 5.4)*.



Slika 5.3: Koraki izvajanja nastavljanja omrežnih naprav in storitev.

Element logičnega modela	Vrednost
Omrežje (*ang. network*)	Omrežje eduroam, ki ima številko vlan 5.
Naslov (*ang. address*)	88.200.67.88/29.
Naprava (*ang. device*)	Osnovni podatki o stikalu sbravos2.
Povezava (*ang. link*)	Povezava od vmesnika TenGigabitEthernet0/2 na stikalu sbravos do vmesnika ge-0/0/23 na stikalu sbravos2.arnes.si.
Vmesnik (*ang. interface*)	Fizični vmesnik za povezavo od sbravos2 do sbravos.
Vmesnik	Upravljalski vmesnik na stikalu sbravos2 z naslovom IP 178.172.79.204/29.
Vmesnik	Privzeti prehod za omrežje eduroam na stikalu lbravos-tsp z naslovom IP 88.200.67.89/29.
Vlan	Nastavitev vlanov na fizičnem vmesniku med sbravos2 in sbravos.
Vlan	Dodatek vlana eduroam na fizičnem vmesniku med lbravos-tsp in sbravos.
Vlan	Dodatek vlana eduroam na fizične vmesnike stikala sbravos2.

Tabela 5.1: Spremembe v logičnem modelu inventorija.

```
GET /api/v1/inventory/
```

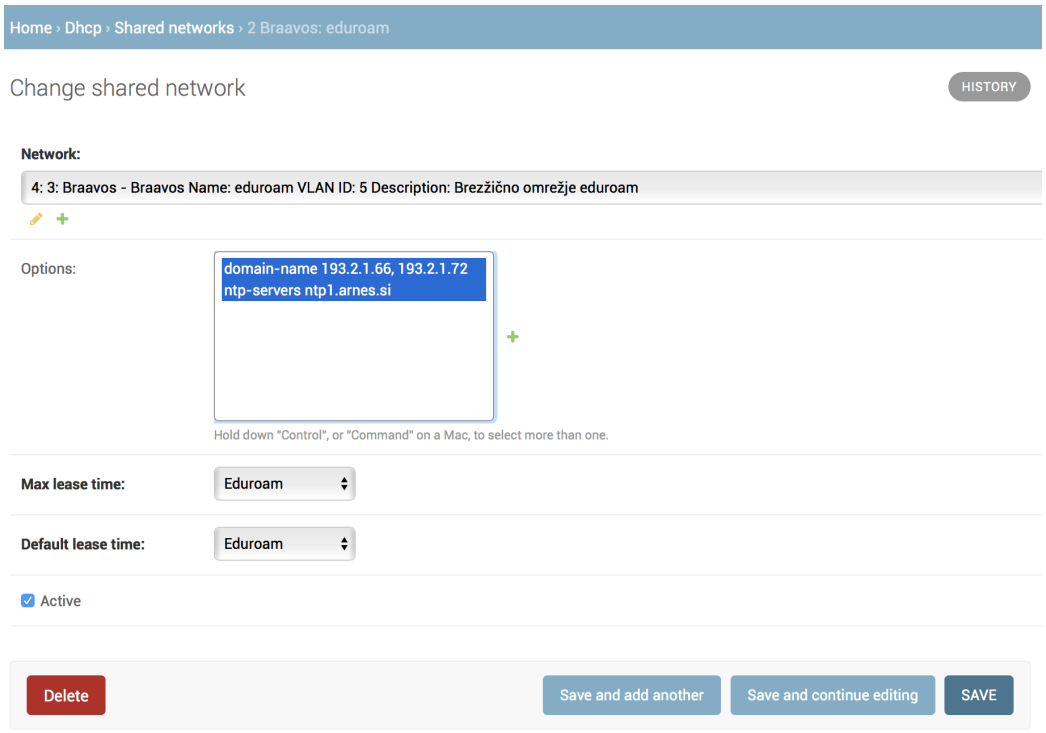
```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "addresses": "http://0.0.0.0:1337/api/v1/inventory/addresses/",
    "interfaces": "http://0.0.0.0:1337/api/v1/inventory/interfaces/",
    "links": "http://0.0.0.0:1337/api/v1/inventory/links/",
    "networks": "http://0.0.0.0:1337/api/v1/inventory/networks/",
    "vlans": "http://0.0.0.0:1337/api/v1/inventory/vlans/",
    "devices": "http://0.0.0.0:1337/api/v1/inventory/devices/",
    "device_models": "http://0.0.0.0:1337/api/v1/inventory/device_models/"
}
```

Slika 5.4: Nabor programskih vmesnikov API v inventoriju.

2. Sistem za upravljanje *(gl. slika 4.4)* je o spremembah v inventoriju obveščen preko sporočilne vrste. Sistem se ob prejemu sporočila avtomatsko poveže na stikala: *lbravos-tsp*, *sbravos*, *sbravos2* in nanje naloži konfiguracije, ki jih pripravi na podlagi podatkov vnešenih v inventoriju. Ker je stikalo sbravos2 nov omrežni element v omrežju, sistem od njega pridobi podatke, kot so serijska številka, model, verzija programske opreme itd. S pridobljenimi podatki sistem dopolni nabor podatkov v logičnem modelu, ki smo jih vnesli mi in nam tako z avtomatskim uvozom skrajša čas za vnos.

3. Po uspešni uveljavitvi sprememb v topologiji omrežja vključimo še storitev DHCP. Vklop izvedemo s preprosto izbiro omrežja v spletnem vmesniku *(gl. slika 5.5)*. Dodatnih parametrov storitve nam ni potrebno nastavljati. Ustrezne nastavitve določi sistem za upravljanje, na podlagi registriranega naslovnega prostora v inventoriju. Nastavitve storitve DHCP sistem zopet strukturirano zapiše v logični model. Zapisane vrednosti so prikazane v tabeli 5.2.



Slika 5.5: Vklop storitve DHCP za omrežje eduroam.

Element logičnega modela	Vrednost
Storitev DHCP (*ang. shared network*)	Vklop storitve DHCP.
Podomrežje (*ang. subnet*)	Podomrežje, v katerem bodo dodeljeni naslovi IP - 88.200.67.88/29.
Nabor naslovov (*ang. pool*)	Nabor naslovov za odjemalce 88.200.67.90 - 88.200.67.94.
Posrednik zahtevkov DHCP (*ang. dhcp relay*)	Vklop posrednika zahtevkov DHCP na stikalu lbravos-tsp, ki je privzeti prehod za omrežje eduroam.

Tabela 5.2: Spremembe v logičnem modelu inventorija in storitve DHCP.

4. Sistem za upravljanje prejme sporočilo, da je prišlo do spremembe v logičnem modelu storitve DHCP. Na podlagi vnešenih podatkov avtomatsko posodobi nastavitve strežnika ISC DHCP. Spodaj je prikazan izsek pripravljene konfiguracijske datoteke za storitev DHCP.

```
# ------------------------------------------------------------ #
#DHCP
#Generated by Automator DHCP Service
# ------------------------------------------------------------ #

#NetworkID: 2
#NetworkName: 4_eduroam
#Tenant: Braavos
#Tenant Identifier: 27883

shared-network "4_eduroam" {

  authoritative;
  default-lease-time 360;
  max-lease-time 360;
  option default-url "4_eduroam";

  subnet 88.200.67.88 netmask 255.255.255.248 {
    option routers 88.200.67.89;
    pool {
      range 88.200.67.90 88.200.67.94;
    }
  }
}
```

5. Odjemalec na sliki se preko vmesnika na stikalu sbravos2 poveže v omrežje eduroam. Z uporabo protokola DHCP pridobi naslov IP in s tem dostop do privzetega prehoda v omrežju.

```
[root@vm4 ~]# ifconfig eth5
eth5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 88.200.67.90 netmask 255.255.255.248 broadcast 88.200.67.95
      inet6 fe80::5054:ff:fe9c:9dcf prefixlen 64 scopeid 0x20<link>
      ether 52:54:00:9c:9d:cf txqueuelen 1000 (Ethernet)
      RX packets 1512471 bytes 509591938 (485.9 MiB)
      RX errors 0 dropped 595 overruns 0 frame 0
      TX packets 1121754 bytes 106387650 (101.4 MiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Prikaz preizkusa dostopa do privzetega prehoda, z uporabo ukaza ping,

za podomrežje je uspešen.

```
[root@vm4 ~]# ping 88.200.67.89
PING 88.200.67.89 (88.200.67.89) 56(84) bytes of data.
64 bytes from 88.200.67.89: icmp_seq=1 ttl=255 time=1.12 ms
64 bytes from 88.200.67.89: icmp_seq=2 ttl=255 time=0.857 ms
```

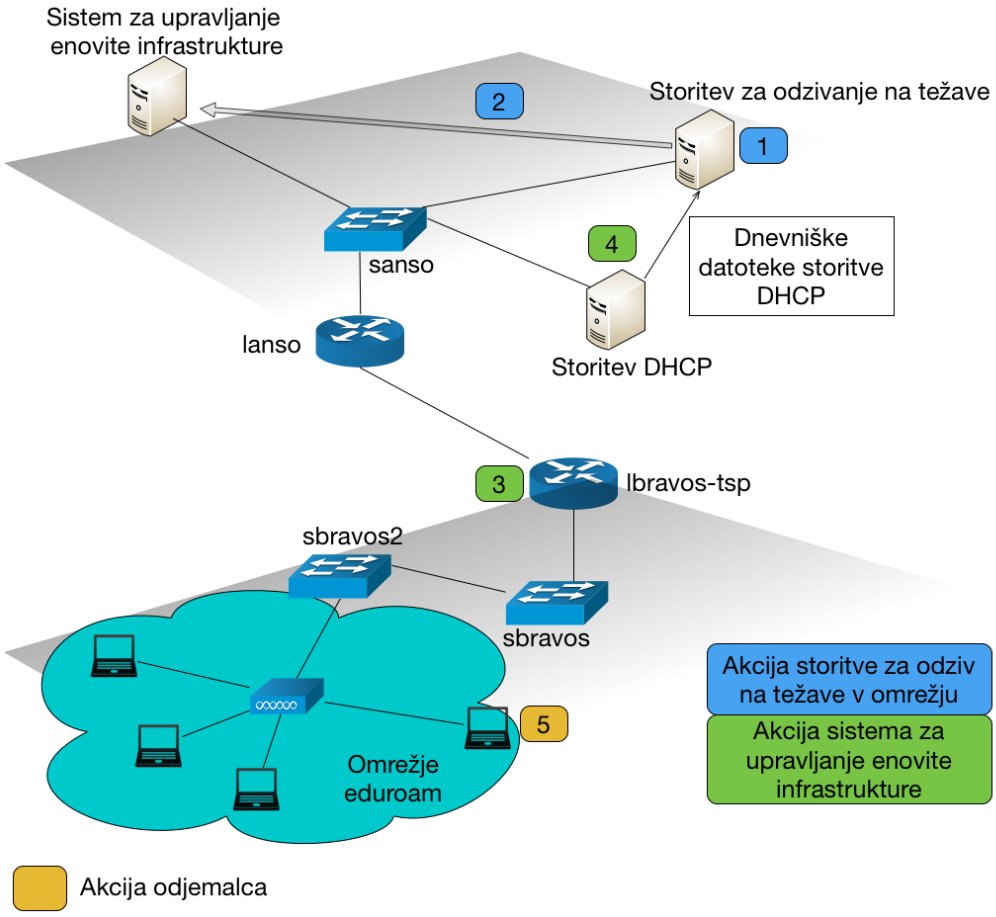
5.2.2 Odzivanje na težave v omrežju

V drugem scenariju smo enovito infrastrukturo uporabili za spreminjanje nastavitev v omrežju na podlagi podatkov stanja v omrežju *(gl. slika 5.6)*. Za potrebe nadzora smo med storitve dodali storitev za odzivanje na težave v omrežju. Ta je sestavljena iz sklada tehnologij Elastic stack [25] in orodja Elastalert [33], ki smo ju predstavili v odstavku 5.2.2. Skupaj omogočata procesiranje dnevniških datotek omrežnih elementov in storitev ter odzivanje na anomalije in težave. Storitev smo razvili kot vlogo za Ansible, saj se njene nastavitve ne spreminjajo pogosto, prav tako je privzeto vključena za vse, ki uporabljajo storitev samodejnega dodeljevanja naslovnega prostora IP. Postopek za dodajanje novih storitev smo opisali v razdelku 4.2.

Elastic stack in Elastalert Elastic stack [25] je sestavljen iz štirih sestavnih delov, ki skupaj omogočajo zbiranje podatkov iz več virov. Nudijo funkcije analize in iskanja po podatkih ter njihovo vizualizacijo v resničnem času. Ti sestavni deli so:

- *Logstash* omogoča zbiranje in procesiranje podatkov,
- *Elastic search* je distribuiran sistem za analizo in iskanje po podatkih,
- *Beats* omogoča pošiljanje podatkov v Logstash ali Elastic search,
- *Kibana* je uporabniški vmesnik za vizualizacijo podatkov.

Njihove funkcionalnosti lahko razširimo z uporabo orodja Elastalert [33], ki omogoča obveščanje o dogodkih na podlagi analize dnevniških datotek v sistemu Elastic search.



Slika 5.6: Koraki izvajanja odzivov na probleme v omrežju.

Storitev za odziv na težave v omrežju periodično bere dnevniške datoteke strežnika za dodeljevanje naslovov IP in zazna, v kolikor na omrežju pride do pomanjkanja naslovnega prostora. Če je obseg dogodka dovolj velik in se ponavlja, aplikacija sproži postopek dodeljevanja dodatnega naslovnega prostora za omrežje s pomanjkanjem naslovov. Aplikacija to stori tako, da v registru naslovnega prostora (*IPAM - IP Address management*) za omrežje ARNES pridobi prosti naslovni prostor in nato spremembe sporoči sistemu za upravljanje enovite infrastrukture. Postopek preizkusa je v obliki korakov predstavljen v spodnjih točkah *(gl. slika 5.6)*.

Element logičnega modela	**Vrednost**
Naslov (*ang. address*)	193.2.194.249/29
Naslov	193.2.194.248/29
Vmesnik (*ang. interface*)	Dodaten naslov 193.2.194.249/29 v polju naslovov.

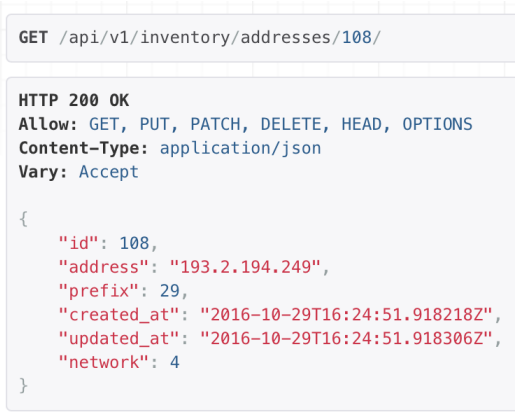
Tabela 5.3: Spremembe v logičnem modelu.

1. Storitev za odzivanje na težave v omrežju se z uporabo orodja Elastalert odzove, ko zazna, da je prišlo do pomanjkanja naslovnega prostora v omrežju eduroam. To se je zgodilo zaradi povečanega števila uporabnikov, saj je brezžično omrežje na organizaciji Bravos postalo zelo popularno pri izvajanju pedagoškega procesa. Spodaj je prikazana dnevniška datoteka strežnika ISC DHCP, v kateri je zabeleženo, da je v omrežju eduroam prišlo do pomanjkanja naslovnega prostora.

```
client: 88.200.67.91, mac: 00:16:36:93:e6:35, shared network: 4_eduroam
DHCPREQUEST for 88.200.67.91 from 00:16:36:93:e6:35 (vm4) via eth1
DHCPACK on 88.200.67.91 to 00:16:36:93:e6:35 (vm4) via eth1
DHCPDISCOVER from 52:54:00:9c:9d:cf via 88.200.67.89: network 4_eduroam: no free leases
```

2. Pravilo v Elastalertu se odzove in sproži skripto *remedy.py*, ki iz registra naslovnega prostora pridobi dodatni naslovni prostor za omrežje eduroam in sistemu sporoči ustrezne spremembe, da se bo omrežna infrastruktura ustrezno prenastavila. Potrebne spremembe nastavitev sistemu pošlje z uporabo programskega vmesnika API *(gl. slika 5.7)*, preko katerega se spremembe shranijo v logični model. Z vmesnikom */api/v1/inventory/addresses/* registrira nov naslovni prostor in z vmesnikom */api/v1/inventory/interfaces/* nastavi privzeti prehod za omrežje *eduroam*. V tabeli 5.3 so prikazane vrednosti, ki se zapišejo v logični model. Spodnji izpis prikazuje izvajanje postopka.

```
INFO:elastalert: Zaganjam
INFO:elastalert: Poizvedba za pravilo odprava pomanjkanja naslovnega prostora IP.
Pomanjkanje naslovnega prostora.
od 2016-10-28 14:33 CEST do 2016-10-29 14:43 CEST: 1 zadetek
Zaganjam odziv na tezavo
```



```
GET /api/v1/inventory/addresses/108/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 108,
    "address": "193.2.194.249",
    "prefix": 29,
    "created_at": "2016-10-29T16:24:51.918218Z",
    "updated_at": "2016-10-29T16:24:51.918306Z",
    "network": 4
}
```

Slika 5.7: Registracija novega naslova IP za omrežni prehod.

```
Pomanjkanje naslovnega prostora IPv4 za omrezje 4_eduroam
Alociram nov naslovni prostor: 193.2.194.248/29
Registriram nov naslovni prostor: 193.2.194.248/29 za omrezje network 4_eduroam
Registriram naslov za privzeti prehod 193.2.194.249/29 v omrezju 4_eduroam
Nastavljam omrezni element: lbravos-tsp.arnes.si -> usmerjevalnik za omrezje: 4_eduroam
Odziv na tezavo zakljucen
INFO:elastalert:Posiljam obvestilo na naslov ['blaz@arnes.si']
INFO:elastalert: Izvedlo se je pravilo za odprava pomanjkanja naslovnega prostora IP.
Pomanjkanje naslovnega prostora. od 2016-10-29 14:32 CEST
do 2016-10-29 14:33 CEST: 1 zadetek pri poizvedbi, 1 ujemanje, 2 obvestili poslani
INFO:elastalert:Spim 58 sekund
```

3. Na podlagi vnešenih sprememb sistem za upravljanje enovite infrastrukture avtomatsko sproži spremembe nastavitev na usmerjevalniku organizacije Bravos z imenom *lbravos-tsp*.

4. Dodatno sistem nastavi tudi storitev DHCP, kjer ji v omrežju eduroam organizacije Bravos vključi dodeljevanje naslovov za dodatno podomrežje.

5. Odjemalec na sliki po nekaj neuspešnih poskusih uspešno pridobi naslov IPv4 iz novo registriranega podomrežja. Spodaj je prikazan izpis vmesnika na odjemalcu.

```
[root@vm4 ~]# ifconfig eth5
eth5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
```

```
inet 193.2.194.250 netmask 255.255.255.248 broadcast 193.2.194.255
inet6 fe80::5054:ff:fe9c:9dcf prefixlen 64 scopeid 0x20<link>
ether 52:54:00:9c:9d:cf txqueuelen 1000 (Ethernet)
RX packets 114231 bytes 7329734 (6.9 MiB)
RX errors 0 dropped 595 overruns 0 frame 0
TX packets 3660 bytes 652225 (636.9 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Preizkus dostopa do privzetega prehoda, z uporabo ukaza ping, za podomrežje je uspešen.

```
[root@vm4 ~]# ping 193.2.194.249
PING 193.2.194.249 (193.2.194.249) 56(84) bytes of data.
64 bytes from 193.2.194.249: icmp_seq=1 ttl=255 time=1.01 ms
64 bytes from 193.2.194.249: icmp_seq=2 ttl=255 time=0.977 ms
```

5.3 Ovrednotenje

S primeroma, ki smo ju predstavili v razdelkih 5.2.1 in 5.2.2, smo prikazali uporabnost predlagane arhitekture. V pričujočih odstavkih smo izpostavili glavne lastnosti predlagane arhitekture in jo ovrednotili na podlagi zahtev upravljalcev, ki so definirane v razdelku 1.1.

Logični model in programski vmesniki API *(gl. razdelek 4.1.2)* so sestavni deli rešitve, zasnovani z namenom poenostaviti upravljanje in upravljalcu omogočiti, da za izvajanje sprememb potrebuje čim manj znanja o omrežjih.

Poenostavitev upravljanja smo dosegli tako, da smo omogočili spreminjanje izbranih nastavitev z uporabo vmesnikov REST. Predlagali smo, da se vmesnike REST razvije le za spreminjanje nastavitev, ki se pogosto spreminjajo, in nastavitev, ki jih želimo izpostaviti uporabnikom. V primerih smo prikazali, kako lahko človeški upravljalec spremembe izvede preko spletnega vmesnika in kako lahko določene akcije avtomatizira s pomočjo skript. V obeh primerih se človeškemu upravljalcu ni potrebno vedeti, kako se bo sprememba izvedla. Njegova naloga je le, da vnese želene spremembe. Izpostavljamo prednost dodatnega logičnega modela, ki je preprostejši od pred-

stavitve omrežja na nivoju abstrakcije, ki je opisan v naslednjem razdelku. Nivo abstrakcije je namreč še vedno zelo tehničen in od uporabnika zahteva precej znanja o omrežjih.

Zavedamo se, da je lahko priprava logičnega modela dolgotrajna, s tem namenom smo v razdelku 4.2 predlagali proceduro za postopno uvedbo novih storitev v omrežje. Procedura predlaga, da se logični model pripravi le za storitve, katerih nastavitve se pogosto spreminjajo in katerih upravljanje želi upravljalec neposredno omogočiti tudi uporabnikom.

Programska oprema zagotavlja samodejno obdelavo sprememb in njihovo uveljavitev na omrežnih elementih smo dosegli, da se človeškemu operaterju ni potrebno zavedati, kje vse se morajo spremembe uveljaviti. Za zanesljiv prenos na prava mesta poskrbita nastavitveni demon in ogrodje Ansible.

Abstrakcijo nastavitev smo zasnovali z namenom spreminjanja nastavitev omrežih elementov in storitev, brez operaterjevega zavedanja sintakse in semantike nastavitvenih podatkov, ki jih ta zahteva. To močno zmanjša napake pri vnosu podatkov in nekonistentnost v nastavitvah. Do napak z uporabo predlagane arhitekture večinoma prihaja med implementacijo novih funkcionalnosti in njihovim testiranjem, saj se računalnik vedno zmoti na enak način. Ko napake odpravimo, se bo postopek vedno izvedel pravilno. Abstrakcija nastavitvenih podatkov zagotavlja tudi, da so ti neodvisni od proizvajalca omrežne naprave. Pri tem smo z uporabo modelov OpenConfig želeli zagotoviti, da so uporabljeni nastavitveni modeli sprejeti v široki skupnosti. Pri tem smo naleteli na slabo podporo v omrežnih elementih, ki smo jih imeli v laboratoriju. Ugotavljamo, da je glavni razlog novost modelov OpenConfig in splošna počasnost proizvajalcev pri uporabi interoperabilnih podatkovnih modelov v svojih napravah. Težavo smo uspešno rešili z uporabo predlog Jinja2, ki instance modelov OpenConfig prevede v nastavitvene podatke, ki jih razume posamezen omrežni element. Priprava predlog zahteva veliko časa in dobro poznavanje posameznega omrežnega elementa. Podpora modelom bo zato močno poenostavila implementacijo novih funkcionalnosti,

saj se bo prevajanje lahko opustilo.

Upravljanje omrežnih naprav in storitev je zahteva, ki smo jo izpolnili z integracijo orodja Ansible v naši rešitvi. Orodje Ansible je predstavljeno v razdelku 3.6. Uporaba orodja Ansible omogoča upravljanje tako omrežnih naprav kot tudi strežnikov, česar v osnovi ne omogočata ne tradicionalno upravljanje omrežij ne krmilniki za programsko osredotočena omrežja. Z uporabo knjižnice NAPALM smo dodatno podprli širši nabor omrežne opreme, ki ne podpira naprednih protokolov, kot je NETCONF. Med raziskavo področja upravljanja računalniških omrežij in storitev smo namreč spoznali, da je podpora upravljalskim protokolom na omrežni opremi še vedno zelo heterogena. Uporaba enega sistema za upravljanje celotne omrežne infrastrukture, omrežnih naprav in strežnikov se močno zmanjša breme vzdrževanja različnih sistemov, s katerim se običajno srečujejo upravljalci omrežij.

Dolgoročna uporabnost rešitve je pomembna lastnost sistema za upravljanje. S tem namenom smo v razdelku 3 raziskali obstoječe rešitve, s pomočjo katerih bi lahko zasnovali arhitekturo rešitve. Z uspešno vključitvijo rešitev, ki so predstavljene v razdelku 3.2, smo zagotovili dobro izhodišče za nadaljnji razvoj in črpanje rešitev iz skupnosti ter prispevanje novih rešitev, ki jih lahko uporabljajo tudi drugi upravljalci.

Dodatno smo v razdelku 4.2 predlagali postopek za razvoj novih funkcionalnosti. Podpora za postopno razširjanje sistema omogoča, da upravljalci hitreje pričnejo z uporabo predlagane arhitekture. To predstavlja odlično alternativo dolgemu načrtovanju, kako v omrežno infrastrukturo uvesti avtomatizacijo in orkestracijo nastavljanja in upravljanja.

V primerih smo prikazali razvoj storitve DHCP. To je mogoče upravljati z uporabo vmesnikov REST in spreminjanje tako lahko omogočimo uporabnikom. Razvoj takšne storitve je kompleksnejši, saj zahteva odločitev, spreminjanje katerih nastavitev želimo izpostaviti preko vmesnikov REST. Na drugi strani smo za drugi primer dodali storitev za odzivanje na težave, ki omogoča zagotavljanje najvišje kakovosti omrežnih storitev. Njene nasta-

vitve se ne spreminjajo pogosto, zato smo jo implementirali le kot vlogo v ogrodju Ansible in jo avtomatično vključili za storitev DHCP. V primeru, da bi se izkazalo, da želimo določen del narediti bolj nastavljiv, lahko naknadno še vedno implementiramo vmesnike REST in omogočimo dinamično konfiguriranje. Menimo, da se takšen pristop k razvoju novih funkcionalnosti dobro sklada z modernimi pristopi k razvoju programske opreme (npr. Scrum, Kanban).

Razširljivost in uporaba odprtokodnih rešitev predstavljata prednost pred tradicionalnim upravljanjem omrežij, ki je običajno zasnovano le okoli enega protokola, in tudi pred krmilniki SDN, ki jih je zaradi zapletenosti težko razširjati.

5.4 Izhodišče za nadaljne delo

Med načrtovanjem, implementacijo in preizkušanjem infrastrukture smo zbrali nekaj izhodišč, ki lahko služijo kot ideje za nadaljnje delo.

Testno voden razvoj nastavljanja omrežja (*ang. Test Driven Development - TDD*) je pomemben aspekt programskega nastavljanja in upravljanja omrežij, saj je potrebno ob spremembah v programski kodi orodja za avtomatizacijo omrežja te dobro preizkusiti pred uporabo v produkcijskih omrežjih. V primeru hrošča lahko napake na omrežju namreč pripeljejo do širšega izpada omrežja in storitev.

Podpora večim storitvam z uporabo agilnih postopkov razvoja programske opreme. Z dodatnimi funkcionalnostmi bi aplikacijam omogočili nadzor nad večjim številom funkcionalnosti omrežja.

Proaktivno odzivanje na težave in anomalije v omrežju , ki smo ga prikazali v na primeru pomanjkanja naslovnega prostora, smo že prikazali možnosti, kako se lahko predlagana infrastruktura odzove glede na zbrane podatke o operativnem stanju omrežja. Z razširitvijo zajetih podatkov in

podatkovnim rudarjenjem obstaja veliko možnosti za razširitev teh funkcionalnosti.

Priprava zbirke predlog za omrežne elemente bo omogočila podporo večjemu naboru omrežnih elementov. Pri implementaciji modelov OpenConfig v našo rešitev smo se namreč zavedli, da jih verjetno nikoli ne bodo podpirali vsi omrežni elementi. Kot rešitev smo uporabili prevajanje nastavitev s predlogami Jinja2. Pri tem smo ugotovili, da se s podobnimi težavami srečuje večina upravljalcev omrežij, zato bi bila dobra rešitev odprtokodni projekt, ki bi obsegal predloge za vse operacijske sisteme, ki jih pogosto najdemo na omrežnih napravah. Slabosti obeh smo izpostavili že v razdelku 2.2.

Poglavje 6

Sklepne ugotovitve

V zadnjih letih se mora omrežna infrastruktura vse bolj prilagajati storitvam. Uporabniki želijo do storitev dostopati na samopostrežen način, kjer je čas od naročila do uporabe storitve kratek. To od omrežne infrastrukture zahteva prilagajanje in nov pristop k upravljanju.

Za vpeljavo programskega upravljanja in nastavljanja omrežne infrastrukture smo predlagali model enovite infrastrukture. Model smo implementirali in preizkusili na laboratorijskem omrežju. Primera uporabe smo zasnovali na podlagi zahtev upravljalcev velikih računalniških omrežij.

Med našim delom smo identificirali nekatere še nerešene izzive, ki dodatno otežujejo enovito upravljanje omrežne infrastrukture. Pomembno odprto vprašanje pri upravljanju omrežne infrastrukture je določitev in sprejetje skupnih modelov za predstavitev nastavitev in operativnega stanja omrežnih naprav. V magistrskem delu smo identificirali dve skupini, ki trenutno pripravljata modele YANG, to sta delovna skupina v IETF in iniciativa OpenConfig. Podpora za modele na omrežne elemente šele prihaja. Kljub vsemu smo v naši rešitvi uporabili modele OpenConfig in njihove instance v nastavitve prevajali z uporabo predlog Jinja2. Njihova uporaba se nam je zdela nujna za dolgoročno uporabnost predlagane arhitekture.

Heterogenost operacijskih sistemov, ki tečejo na omrežnih elementih, prinaša različno podporo metodam in protokolom za upravljanje. Z raziskavo in ovre-

dnotenjem metod smo ugotovili, da ni enega samega protokola, ki bi nam zagotovil enovito upravljanje, zato smo v svoji rešitvi uporabili knjižnico NAPALM, ki omogoča upravljanje omrežnih elementov z uporabo različnih protokolov.

Menimo, da mora biti sprememba pristopa k upravljanju izvedena postopoma. S tem namenom smo v razdelku 4.2 podali predlog postopka za razvoj novih storitev. Postopek bi zagotovil hitrejše dodajanje podpore za nove storitve in njihovo postopno uvajanje v sistem za upravljanje.

Ocenjujemo, da se bo z vse večjo potrebo in zahtevami upravljalcev omrežij po programskem upravljanju izboljšala tudi podpora za skupne podatkovne modele. Prav tako bodo omrežne naprave bolje podprle metode za upravljanje, ki so primernejše za programsko upravljanje.

Dodatek A

Izvorna koda in dokumentacija

Izvorna koda rešitve in dokumentacija sta na voljo na repozitoriju git na naslovu https://github.com/blazdivjak/automator.

Literatura

[1] Django. Dostopno na: `http://djangoproject.com`, 2016. Zadnji dostop 1. 10. 2016.

[2] gRPC. Dostopno na: `http://www.grpc.io`, 2016. Zadnji dostop 31. 5. 2016.

[3] OpenConfig. Dostopno na: `http://www.openconfig.net`, 2016. Zadnji dostop 1. 5. 2016.

[4] Opendaylight. Dostopno na: `https://www.opendaylight.org`, 2016. Zadnji dostop 10. 9. 2016.

[5] Python. Dostopno na: `https://www.python.org`, 2016. Zadnji dostop 1. 10. 2016.

[6] The Benefits of Agentless Architecture. White Paper. Dostopno na: `https://www.ansible.com/benefits-of-agentless-architecture`, 2016. Zadnji dostop 15. 9. 2016.

[7] YANG Central. Dostopno na: `http://www.yang-central.org`, 2016. Zadnji dostop 1. 10. 2016.

[8] B. Aboba and D. Thaler. What Makes for a Successful Protocol? RFC 5218, IETF, julij 2008.

[9] I. Al Ajarmeh and J. Yu. An empirical study of the NETCONF protocol. In *Networking and Services (ICNS), 2010 Sixth International Conference on*, pages 253–258. IEEE, 2010.

[10] D. Barroso and E. Jasinska. Network automation and programmability abstraction layer with multivendor support. In *NANOG64*, 2015.

[11] M. Bertolina. NETCONF Element Management System – Design and Implementation. In *Journal of Computer Science and Technology*, volume 12, pages 87–90. Facultad de Informática, avgust 2012.

[12] S. Bhushan and L. Poulopoulos. ncclient. Dostopno na: `http://ncclient.org`, 2016. Zadnji dostop 20. 5. 2016.

[13] A. Bierman. A Guide to NETCONF for SNMP Developers. In *IEEE802 Plenary*, pages 17–24, 2014.

[14] A. Bierman, M. Bjorklund, and K. Watsen. RESTCONF Protocol draft-ietf-netconf-restconf-12. RFC Draft, IETF, oktober 2016.

[15] A. Bierman, J. Schoenwaelder, A. Sehgal, and P. van der Stok. CoAP Management Interface draft-vanderstok-core-comi-10. RFC Draft, IETF, oktober 2016.

[16] M. Bjorklund. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020, IETF, oktober 2010.

[17] M. Bjorklund. The YANG 1.1 Data Modeling Language draft-ietf-netmod-rfc6020bis-07. RFC Draft, IETF, marec 2016.

[18] M. Bjorklund, A. Bierman, R. Enns, and J. Schoenwaelder. Network Configuration Protocol (NETCONF). RFC 6241, IETF, junij 2011.

[19] M. Björklund, J. Schoenwaelder, and P. Shafer. Network configuration management using NETCONF and YANG. *IEEE communications magazine*, 48(9):166–173, 2010.

[20] C. Black and P. Goransson. *Software Defined Networks: A Comprehensive Approach*. Elsevier, 2014.

[21] J. Case, J. Davin, M. Fedor, and M. Schoffstall. A Simple Network Management Protocol (SNMP). RFC 1157, IETF, maj 1990.

[22] T. Christie. Django REST framework. Dostopno na: `http://www.django-rest-framework.org`, 2016. Zadnji dostop 1. 10. 2016.

[23] F. Dijkstra and E. Ruiter. Choosing an open source network controller for a data center network. In K. Wierenga, editor, *TNC15*. GÉANT - formerly TERENA, november 2015.

[24] R. T. Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[25] C. Gormley and Z. Tong. *Elasticsearch: The Definitive Guide*. O'Reilly Media, Inc., 2015.

[26] J. Gray et al. The transaction concept: Virtues and limitations. In *International Conference on Very Large Databases*, pages 144–154, 1981.

[27] D. Hall. *Ansible Configuration Management*. Packt Publishing Ltd, 2013.

[28] IETF netmod working group. NETCONF Data Modeling Language (netmod). Dostopno na: `https://datatracker.ietf.org/wg/netmod/documents/`, 2016. Zadnji dostop 20. 5. 2016.

[29] D. Kreutz, F. Ramos, P. Esteves Verissimo, C.E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[30] L. Lhotka. Working with YANG data models and instances using (mainly) pyang. Dostopno na: `https://www.ietf.org/edu/tutorials/90-YANG-Tutorial.pdf`, 2014. Zadnji dostop 25. 5. 2016.

[31] L. Lhotka and CZ.NIC. JSON Encoding of Data Modeled with YANG draft-ietf-netmod-yang-json-04. RFC Draft, IETF, junij 2015.

[32] L. Lisboa Penz, A. Stone, and D. Venkata Naga Praveen. OpenYuma. Dostopno na: `https://github.com/OpenClovis/OpenYuma`, 2016. Zadnji dostop 20. 5. 2016.

[33] Q. Long et al. ElastAlert. Dostopno na: `https://github.com/Yelp/elastalert`, 2016. Zadnji dostop 10. 10. 2016.

[34] Y. Mitsos and L. Poulopoulos. NETCONFish: speaking the network language. In D. Foster, editor, *TNC13*. TERENA, avgust 2013.

[35] E. Nilsen-Nygaard et al. IETF YANG Models on Github. Dostopno na: `https://github.com/YangModels/yang`, 2016. Zadnji dostop 1. 10. 2016.

[36] H. Porat. Reality Check: SDN – great in theory, less in practice. Dostopno na: `http://www.rcrwireless.com/20160202/opinion/reality-check-sdn-great-in-theory-less-in-practice-tag10`, 2016. Zadnji dostop 11. 11. 2016.

[37] L. Poulopoulos. NETCONF Proxy. Dostopno na: `https://code.grnet.gr/projects/nxpy`, 2016. Zadnji dostop 20. 5. 2016.

[38] L. Richardson and S. Ruby. *RESTful web services*. O'Reilly Media, Inc., 2008.

[39] M. Rose and K. McCloghrie. Structure and Identification of Management Information for TCP/IP-based Internets. RFC 1155, IETF, maj 1990.

[40] T. Saje. Upravljanje in nadzor fakultetne infrastrukture s protokolom SNMP. BSc thesis, Univerza v Ljubljani, 2013.

[41] J. Schoenwaelder. Overview of the 2002 IAB Network Management Workshop. RFC 3535, IETF, maj 2003.

[42] J. Schoenwaelder and F. Strauss. SMIng - Next Generation Structure of Management Information. RFC 3780, IETF, maj 2004.

[43] A. Shaikh and R. Shakir. OpenConfig YANG Models on Github. Dostopno na: `https://github.com/openconfig/public`, 2016. Zadnji dostop 1. 10. 2016.

[44] R. Shakir. pyangbind. Dostopno na: `http://pynms.io/pyangbind/`, 2016. Zadnji dostop 26. 5. 2016.

[45] M. Subramanian. *Network Management: Principles and Practice*. Addison-Wesley, 2000.

[46] J. Turnbull. *Pulling strings with puppet: configuration management made easy*. Apress, 2008.

[47] S. Wallin. Tutorial: NETCONF and YANG. In *NANOG62*, 2013.

[48] S. Wallin and C. Wikstrom. Automating Network and Service Configuration Using NETCONF and YANG. In *LISA*, 2011.